MYD-SAMA5D3X
User Manual

Version V1.0

Version History

Version Number	Description	**Time**
V1.0	Initial Version	2013.05

Directory

Chapter 1 Product Overview 1

1.1 Product Description 1

1.2 Product Preview 1

1.3 Product Features 1

1.4 Product Configuration 5

Chapter 2 Hardware Resource Introduction 6

2.1 CPU module + Base Board Resources Overview 6

2.2 CPU module Introduction 7

2.2.1 CPU module architecture and CPU 7

2.2.2 DDR2 SDRAM 8

2.2.3 Clock Circuit 9

2.2.4 Serial DATAFLASH 10

2.2.5 NORFLASH & NAND-FLASH 11

2.2.6 LED 13

2.2.7 ETHERNET Module 14

2.3 Base Board Introduction 14

2.3.1 Base Board Architecture & power supply 14

2.3.2 CAN 15

2.3.3 RS485 16

2.3.4 HSMCI Module 17

2.3.5 SODIMM Interface 17

2.3.6 Smart DAA Module 20

2.3.7 Audio Module 20

2.3.8 USB Module 21

2.3.9 ETHERNET Module 23

2.3.10 HDMI Module 24

2.3.11 USART Module & DBUG 25

2.4 Jumper Setting 26

Chapter 3 MDK use 27

Chapter 4 Linux System Guide 28

4.1 Outline 28

4.2 Software Resources 29

4.3 Start Linux System 30

4.3.1 Install Download Tool 30

4.3.2 Connect Board to PC 30

4.3.3 Automatic Download 32

4.3.4 Manual Download 32

4.4 Linux Development Environment Structure 41

4.5 Installation and Compile 41

4.5.1 Create a Working Directory 41

4.5.2 Install Cross Compiler Tools 42

4.5.3 Install AT91Bootstrap Source and Compile 42

4.5.4 Install uboot Source and Compile 43

4.5.5 Install and Compile Linux kernel Source Code 43

4.5.6 Install and Compile Linux File System 43

4.6 Make Linux File System 44

4.6.1 Write Demo Program hellomyir 44

4.6.2 Mount UBIFS File System 45

4.6.3 Modify UBIFS System Files 46

4.6.4 Regenerate UBIFS System File 47

4.7 Linux Use 49

4.7.1 Touch Screen Calibration 49

4.7.2 U disk Use 50

4.7.3 SD Card Use 51

4.7.4 Play MP3 Music 52

4.7.5 Network Port Test 52

4.7.6 Telnet Test 54

4.7.7 Login board via telnetd 55

4.7.8 Mount NFS file system 56

4.7.9 RTC Use 57
4.8 Linux Driver Development Examples 57
4.8.1 Hardware Schematic 58
4.8.2 Driver Source Code 58
4.8.3 Compile the Driver 65
4.9 Application Development Instance 66
4.9.1 Download Driver into Board 66
4.9.2 Driving test 67
4.10 Qt use Guide 67
4.10.1 Using the Qt cross compiler tool chain 67
4.10. 2 Qt development environment 71
Chapter 5 Android System Guide 76
5.1 Overview 76
5.2 Software Resources 77
5.3 Build Android System 78
5.3.1 Install Download Tool 78
5.3.2 Connect Board and SAM-BA 78
5.3.3 Automatic Download 79
5.3.4 Manual Download 79
5.4 Compile Android System Files 87
5.4.1 Android System Principle 87
5.4.2 Compile System Files 88
5.5 Android System Use 89
5.5.1 USB Keyboard Test 89
5.5.2 Browse Picture Test 90
5.5.3 Play Audio Test 91
5.5.4 Ethernet Test 92
5.5.5 APK Test 94
Appendix 1 FAQ 95
Appendix 2 sales FAQ and technical support 98

Chapter 1 Product Overview

1.1 Product Description

MYIR has lunched MYD-SAMA5D3X series boards which are based on Atmel ATSAMA5D3Xseries processor (SAMA5D31，SAMA5D33，SAMA5D34，SAMA5D35 based on the ARM Cortex-A5 kernel). Running at up to 536 MHz, MYD-SAMA5D3X have 256MB Nand-Flash, 4MB Data-Flash(16bits),512MB DDR2 SDRAM and supports Linux 3.6.9 as well as Android 4.0.4 operating system, which also provides relevant sources and have rich peripheral interfaces: High-speed USB2.0, Audio input, Audio output, LCD interface, CAN interface, JTAG debug interface, Serial port and Micro SD card interface.

1.2 Product Preview

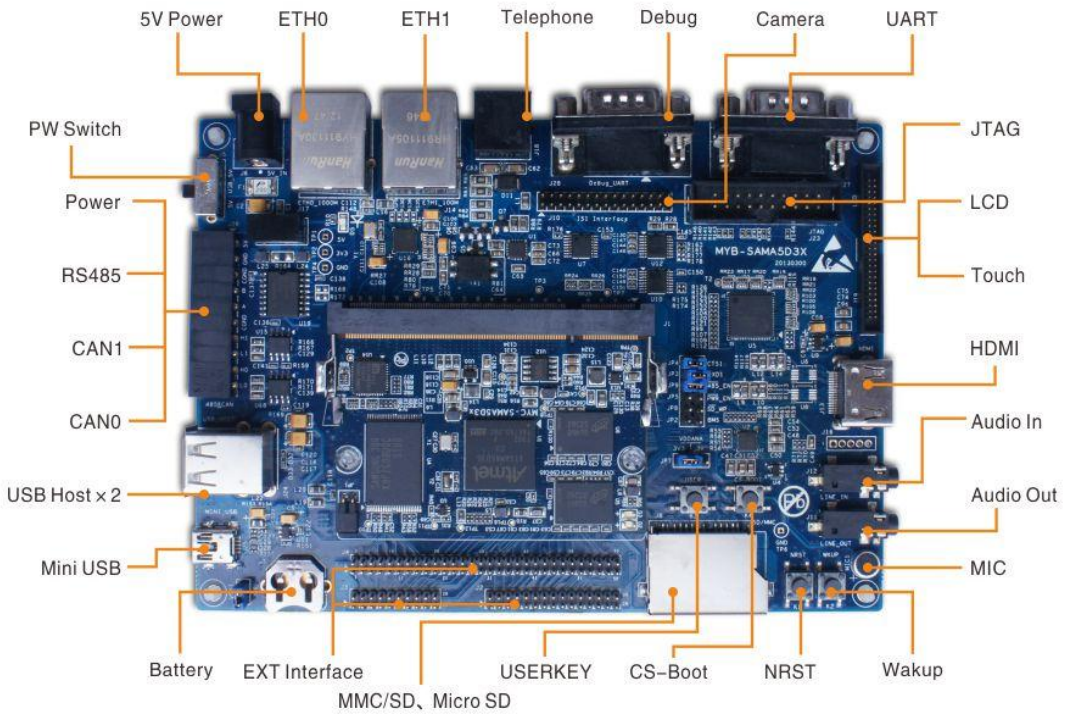


Figure 1-1

1.3 Product Features

Based on ARM Cortex-A5 processor, MYD-SAMA5D3X integrates all the chip functi ons and features. The key features are as follows:

- High-performance architecture — Based on the ARM Cortex-A5 core with floating-point unit, the SAMA5D3 series is ideal for applications requiring high-precision computing and fast data processing. Features a 64-bit internal bus architecture and 32-bit wide DDR controller running up to 166MHz that offers up to 1328MB/s of bandwidth
- Market-leading low power consumption — Ideal for battery- operat ed applications, the SAMA5D3 series consumes less than 200mW when running at maximum speed and under 0.5mW in low-power mode
- Comprehensive peripheral set for connectivity and user interface applications — Includes Gigabit and 10/100/1000 Ethernet, up to three HS USB ports, dual CAN, three SDIO/SD/MMC, UARTs, SPIs, TWIs, soft modem, LCD controller with graphics accelerator, camera interface, 12-bit ADC, 32-bit timers and more
- State-of-the-art security — Includes secure boot, Advanced Encryption Standard (AES)/Triple Data Encryption Standard (DES) encryption engine, Secure Hash Algorithm (SHA) and True Random Number Generator (TRNG)
- Low system cost — The 0.8mm ball pitch package reduces the PCB design complexity. A simple power management scheme and impedance matching on DDR lines reduce the need for external components. Three USB ports save the cost of an external USB hub.

The following simple lists the basic features of MYD-SAMA5D3X

Electrical parameters:

- Working temperature
  - Industrial level Temperature: -40℃~85℃

- non-industrial level temperature : 0℃ - 70℃

- Operating humidity:0 to 90%
- Electrical Specifications: +5V power supply
  - Base Board: +5V power supply
  - CPU module: +3.3V power supply
- Mechanical Dimensions:
  - Base Board: 【154 * 110】 mm
  - CPU module: 【45 * 67.6】 mm

Processor:

- SAMA5D31//D33/D34/D35 (32 bits ARM v7-A Thumb2 processor) runs at up to 536MHz
- 32KB Data Cache, 32KB Instruction Cache

Memory:

- 160KB chip ROM ， 128KB chip SRAM
- 512MB DDR2 SDRAM
- 256MB Nandflash
- 16MB NorFlash(16 bits)
- 4MB DataFlash

Audio and Video Interface:

- An Audio 3.5mm Input Interface
- An MIC input interface
- A Two-channel Audio 3.5mm Output Interface

Transmission Interface:

- Standard JTAG Interface
- Micro SD Card Interface
- SD/MMC Interface
- Serial Ports
  - 1x DBGU Port (Debug Unit)
  - 1xUSART(5 pin)
- Two CAN Interfaces (Only MYD-SAM A5D34 and MYD-SAMA5D35 have CAN

Interface)

- 2 High-speed USB HOST Interfaces
- A Mini USB Host/Device Interface
- Ethernet MAC

processor	SAMA5D31	SAMA5D33	SAMA5D34	SAMA5D35
Ethernet	10/100 M	10/100/1000 M	10/100/1000 M	Dual Ethernet

LED Indicator:

- A Power Indicator (CPU module: Red)
- A Users Light/System Heartbeat Light (CPU module: Blue)
- A Power Indicator (Base Board: Red)

1.4 Product Configuration

No	Name	Number	Note
1	MYD-SAMA5D3X Development Board	1	Base Board+CPU module
2	1.5 Meters Crossover Cable	1	
3	1.5 Meters Mini USB 2.0 Cable	1	
4	5V/2A DC Power adapter	1	
5	3V Button Batteries	1	
6	Serial Cable(DB9-To-DB9 9pins)	1	
7	Product DVD	1	Include Schematic (PDF), User Manual, Source Code, etc.
8	4.3 Inch LCD Touch Screen	1	optional

Chapter 2 Hardware Resource Introduction

2.1 CPU module + Base Board Resources Overview

SAMA5D3X resources are shown in table2-1

Name		5D31	5D33	5D34	5D35
Processor		Atmel SAMA5D31/5D33/5D34/5D35(ARM Cortex-A5 core，Frequency at up to 536MHz)			
Memory		512MB DDR2 SDRAM			
Flash		256MB Nand-Flash, 4MB Data-Flash, 16MB NorFlash(16 Bits)			
USB	USB HOST	2	2	2	2
	Mini USB Host / Device	1	1	1	1
Audio	Audio Input	1	1	1	1
	MIC Input	1	1	1	1
	Audio Output	1	1	1	1
Network	ETH 10/100M Ethernet Port	1	0	0	1
	ETH 1000M Ethernet Port	0	1	1	1
Serial	DBGU Serial	1	1	1	1
	USART1	1	1	1	1
JTAG	JTAG Interface	1	1	1	1
LCD	Support 4.3 Inch Touch Screen	1	1	1	0
RTC	Real Time Clock On	1	1	1	1

	Board and backup battery				
Extended Interface	20 Pins User Extended Interface	1	1	1	1
	30 Pins User Extended Interface	1	1	1	1
	60 Pins User Extended Interface	1	1	1	1
Power	5V Power Input	1	1	1	1
SD Card	Micro SD Interface	1	1	1	1
SD/MMC	SD/MMC Interface	1	1	1	1
CAN	CAN Interface	0	0	2	2
RS485	RS485 Interface	1	1	1	1
Button	User Button	1	1	1	1
	System Button	3	3	3	3

Table 2-1

2.2 CPU module Introduction

2.2.1 CPU module architecture and CPU

(1)CPU module architecture is shown in figure 2-1:

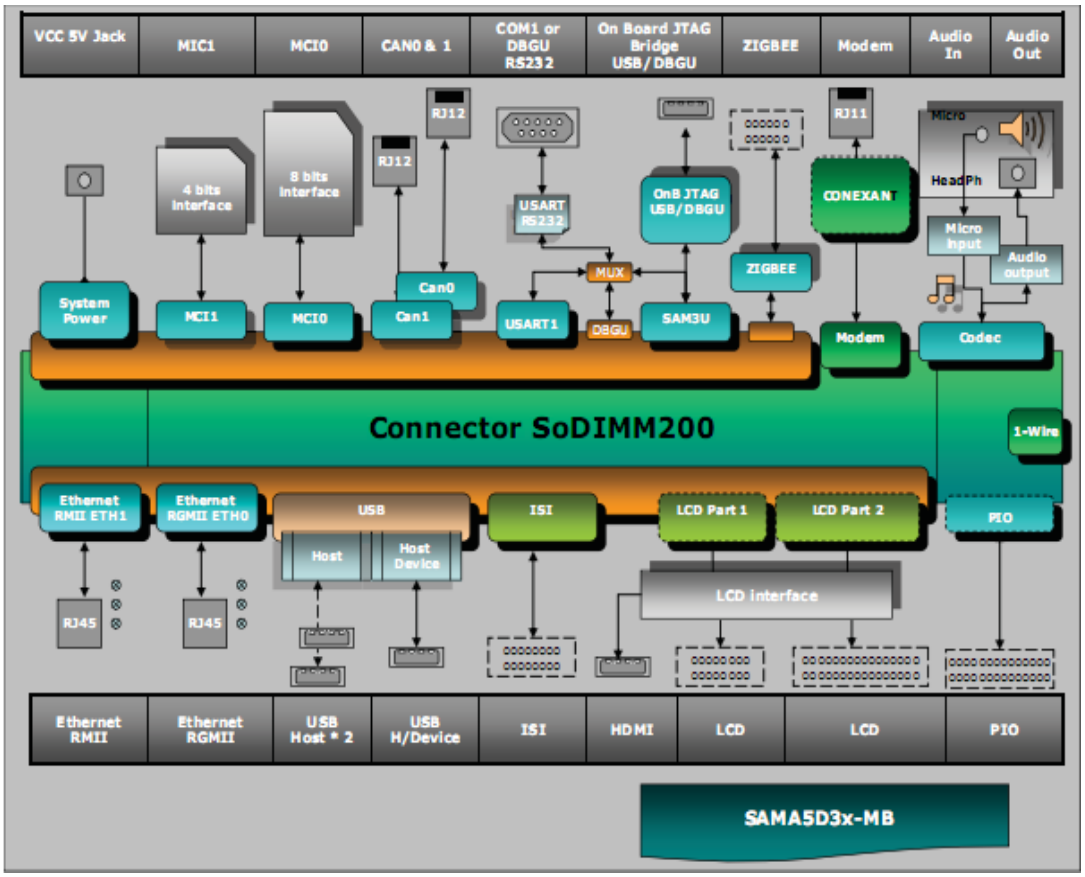


Figure 2-1

(2) CPU

The Atmel SAMA5D3 series is a high-performance, power-efficient embedeed MPU based on the ARM Cortex-A5 processor, achieve 536MHZ with power consumption levels below 0.5mW in low-power mode. The device features a flexible size instruction and data caches, memory management unit(MMU) and Floating-Point Unit (FPU). The Cortex-A5 processer implements the ARMv7 VFPv4-D16 architecture and runs 32-bits ARM instructions,16-bits Thumb instructions and 8-bit Java byte codes in Jazelle state The processor has the ARM ,Thumb ThumbEE, Jazelle states controlled by the T bit and J bit in the CPSR.

2.2.2 DDR2 SDRAM

DDRAM device key features

- VDD=+1.8V±0.1V, VDDQ= +1.8V ±0.1V
- All inputs and outputs are compatible with SSTL_18 interface

- Auto refresh and self-refresh
- Organizational structure:8 blanks, Page size: 128 Meg x 16 (16 Meg x 16 x 8 banks)
- Programmable CAS latency (CL)
- Programmable CAS additive latency (AL)
- Programmable burst lengths:4 or 8
- Adjustable data-output drive strength

DDRAM circuit peripheral is shown in figure 2-2:

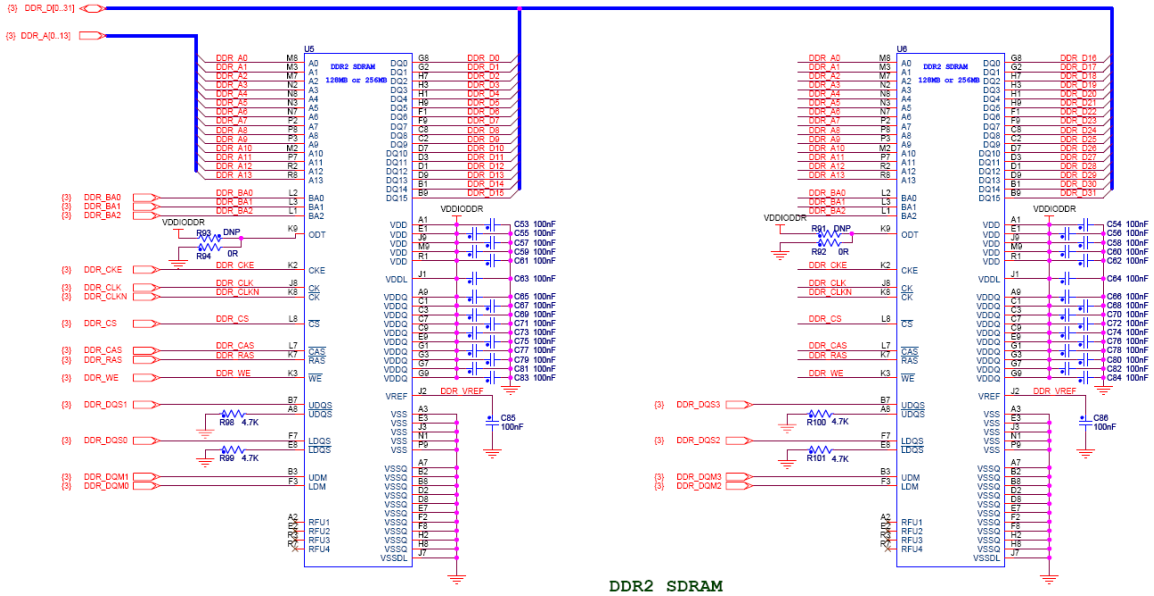


Figure 2-2

2.2.3 Clock Circuit

(1) Internal clock choose 12 MHz crystal. Clock Circuit is shown in figure 2-3:

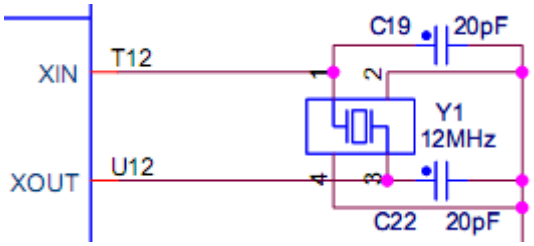


Figure 2-3

(2) RTC clock chooses 32.768 KHz crystals. Circuit is shown in figure 2-4:

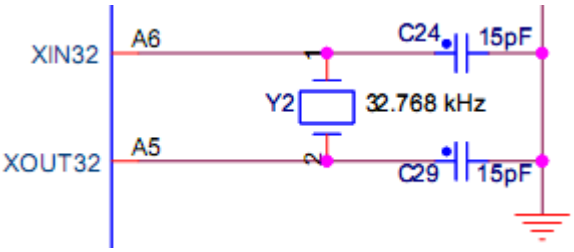


Figure 2-4

2.2.4 Serial DATAFLASH

4MB DATAFLASH, Its characteristics are as follows:

- Single 2.7V-3.6V Supply
- Serial Peripheral interface (SPI) Compatible
  - Support SPI Modes 0 and 3
- 70 MHz Maximum Clock Frequency
- Flexible, Uniform Erase Architecture
  - 4-Kbyte Blocks,32-Kbyte Blocks,64-Kbyte Blocks, Full Chip Erase
- Individual Sector Protection with Global Protect/Unprotect Feature
  - 64-Kbyte Physical Sectors
- Hardware Controlled Locking of Protected Sectors
- Flexible Programming
  - Byte/Page Program(1 to 256 Bytes)
- Automatic Checking and Reporting of Erase/Program Failures
- JEDEC Standard Manufacture and Device ID Read Methodology
- Low Power Dissipation
  - 7 mA Active Read Current (Typical)
  - 15 µA Deep Power-Down Current (Typical)
- Endurance:100,000 Program/Erase Cycles
- Data Retention: 20 Years
- Complies with Full industrial Temperature Range
- Industry Standard Green (Pb/Halide-free/RoHS Compliant) Package Options
  - 8-lead SOIC (200 mil wide)

- 16-lead SOIC (300 mil wide)

Processor has two SPI. SPI0 controls data flash. Refer to figure 2-5:

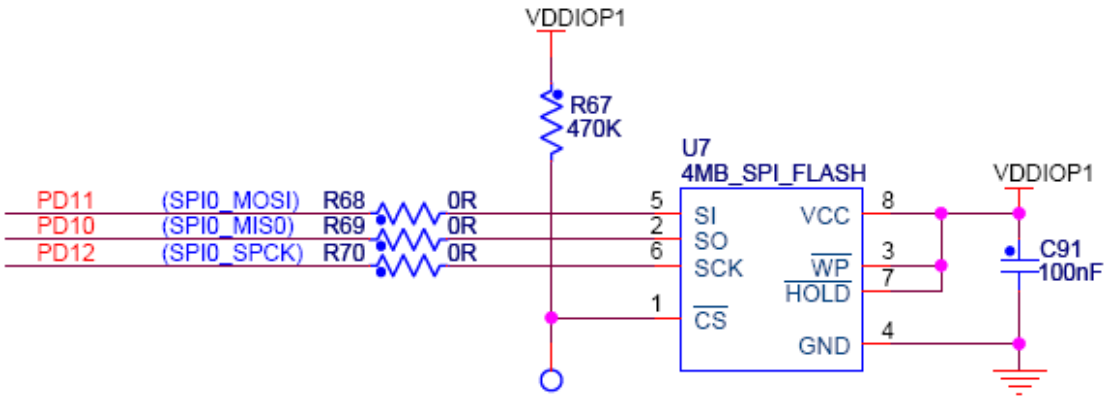


Figure 2-5

2.2.5 NORFLASH & NAND-FLASH

(1) 16MB and its characteristics are as follows:

- Organization:
  - Page size: 128K Bytes(4Blocks top to bottom)
  - Block size: 32k Bytes
- Read Operation:
  - Random Read: 25 us
  - Serial Access: 25 ns
- Fast Write Cycle Time:
  - Page Program time: 200 us(Typ)
  - Block Erase Time: 20 us (Typ)
- Power:
  - VCC (core): 1.7 V – 2.0 V
  - VCCQ (I/O): 1.7 V – 3.6 V
- Endurance: 100,000 Program/Erase Cycles
- Data Retention: 10 Years

NORFLASH circuit is shown in figure 2-6:

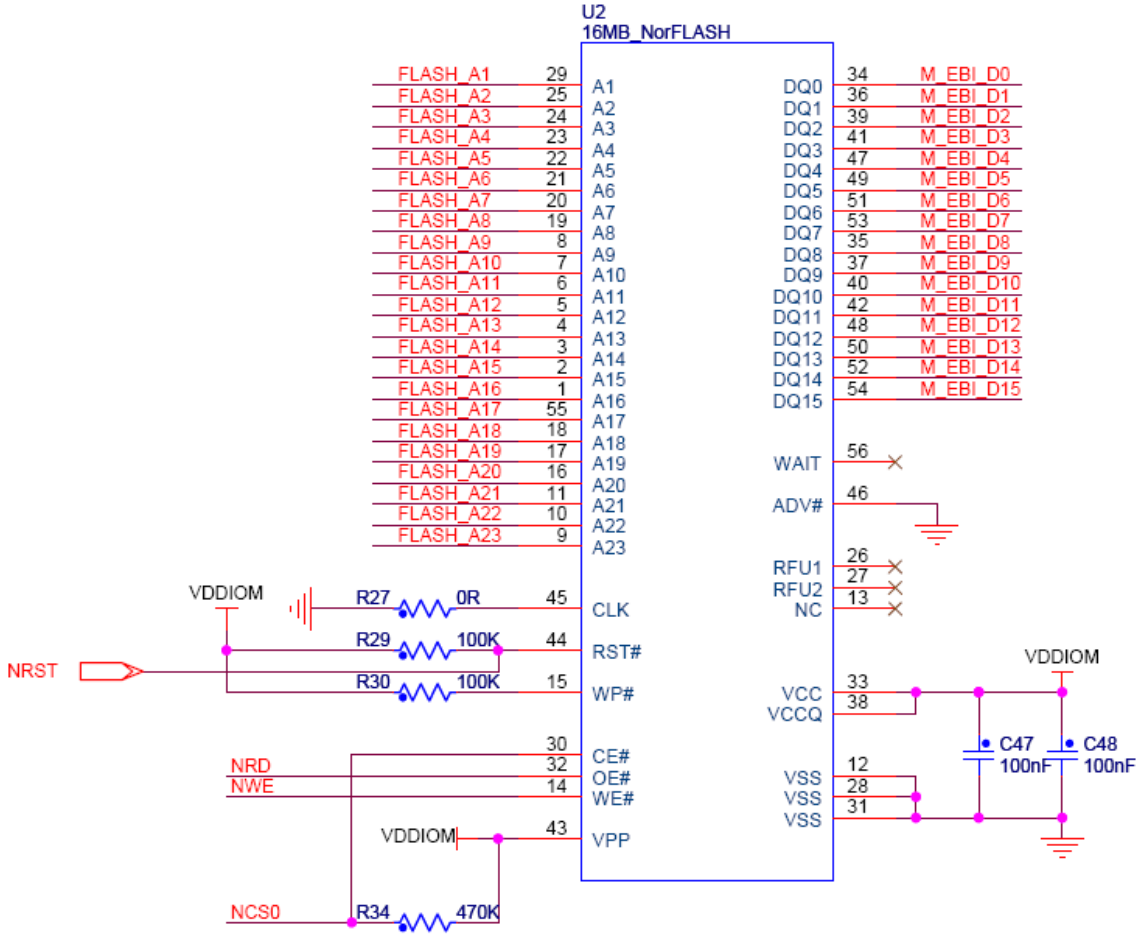


Figure 2-6

(2) 256MB NANDFLASH and its characteristics are as follows:

- Organization:
  - Page size: 2K + 64 Bytes
  - Block size: 128K + 4K Bytes (64 Pages)
  - Plane size: 1056Mb((128K+4K)x1024Blocks)
  - Device size: 2112Mb(1056Mbx2planes)
- Array performance
  - Read page : 25 us
  - Program page:200 us(TYP:1.8v,3.3V)
  - Erase block: 700us(TYP)
- Power: 2.7V–3.6V
- Endurance: 100,000 Program/Erase Cycles
- Data Retention: 10 Years

- 48-pin TSOP type1,CPL package
- WP# signal: Write protect entire device

NAND-FLASH circuit is shown in figure 2-7:

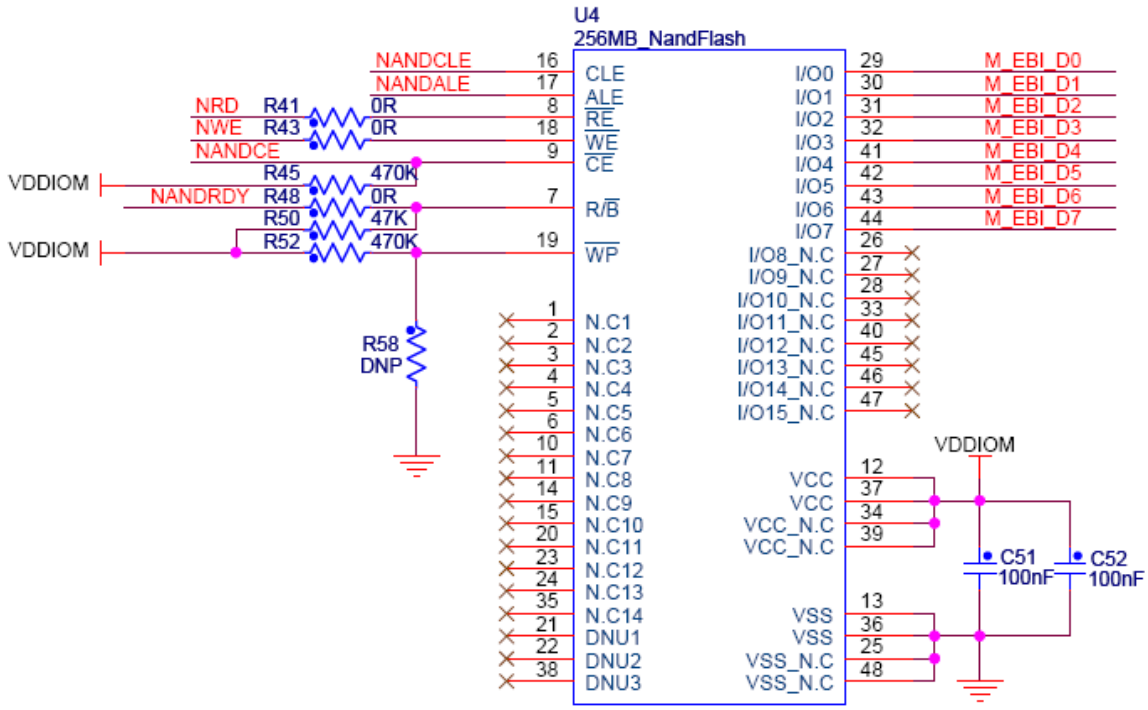


Figure 2-7

2.2.6 LED

System LED and User LED circuits are as shown in figure 2-8:

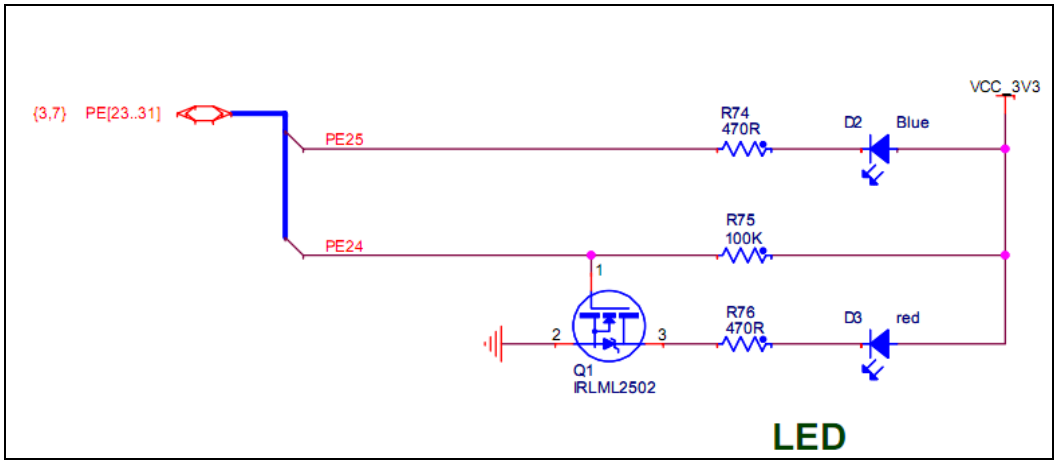


Figure 2-8

2.2.7 ETHERNET Module

ETHERNET's characteristics are as follows:

- Single-chip 10/100/1000Mbps IEEE compliant Transceiver
- RGMII1.3 Interface support
- 25MHZ crystal is used to generate all required clocks
- Power down and power saving modes
- MDO/MDIO Management Interface
- 48-pin (5mm x 5mm) QFN package
- Power supply
  - VCC (core): 1.2 V
  - VCCQ (I/O): 3.3 V /2.5 V

ETHERNET Module is shown in figure 2-9:

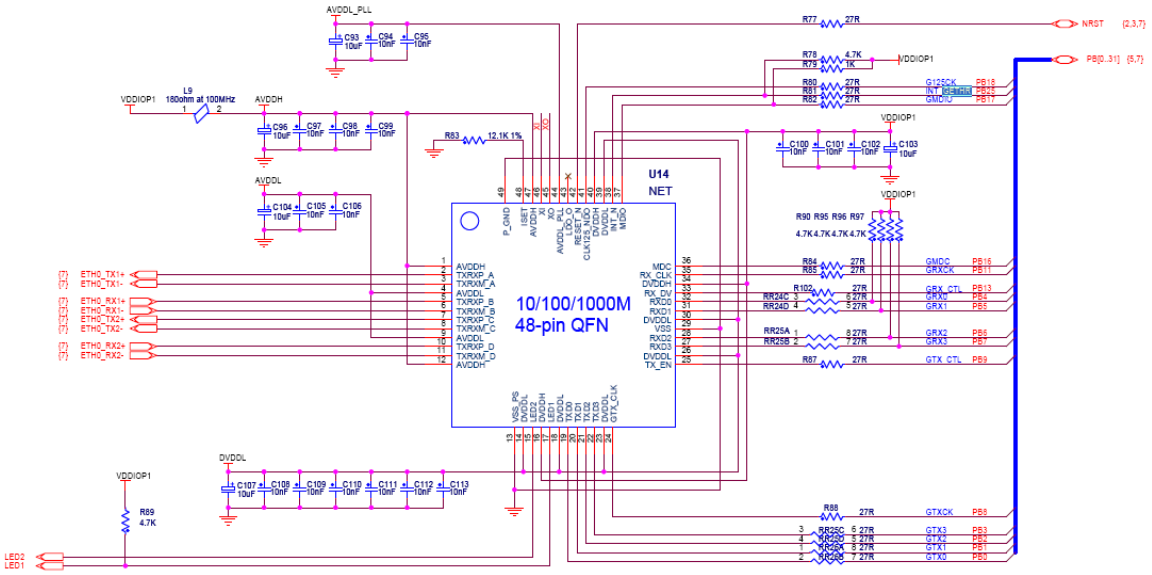


Figure 2-9

2.3 Base Board Introduction

2.3.1 Base Board Architecture & power supply

(1) Device power supply is shown in figure 2-10

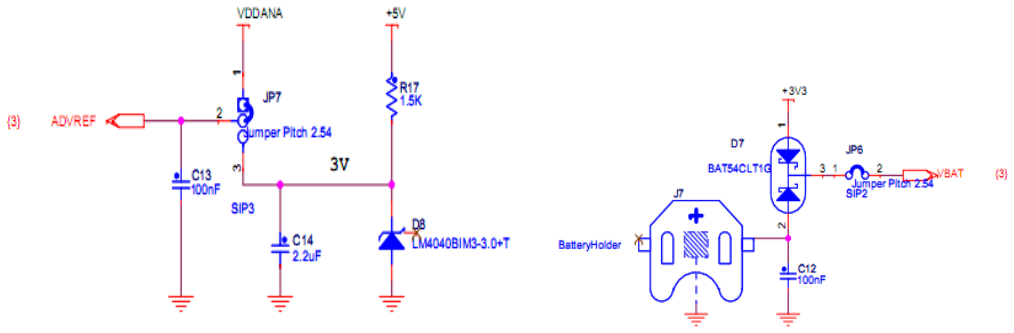


Figure 2-10

(2) Push button is shown in figure 2-11

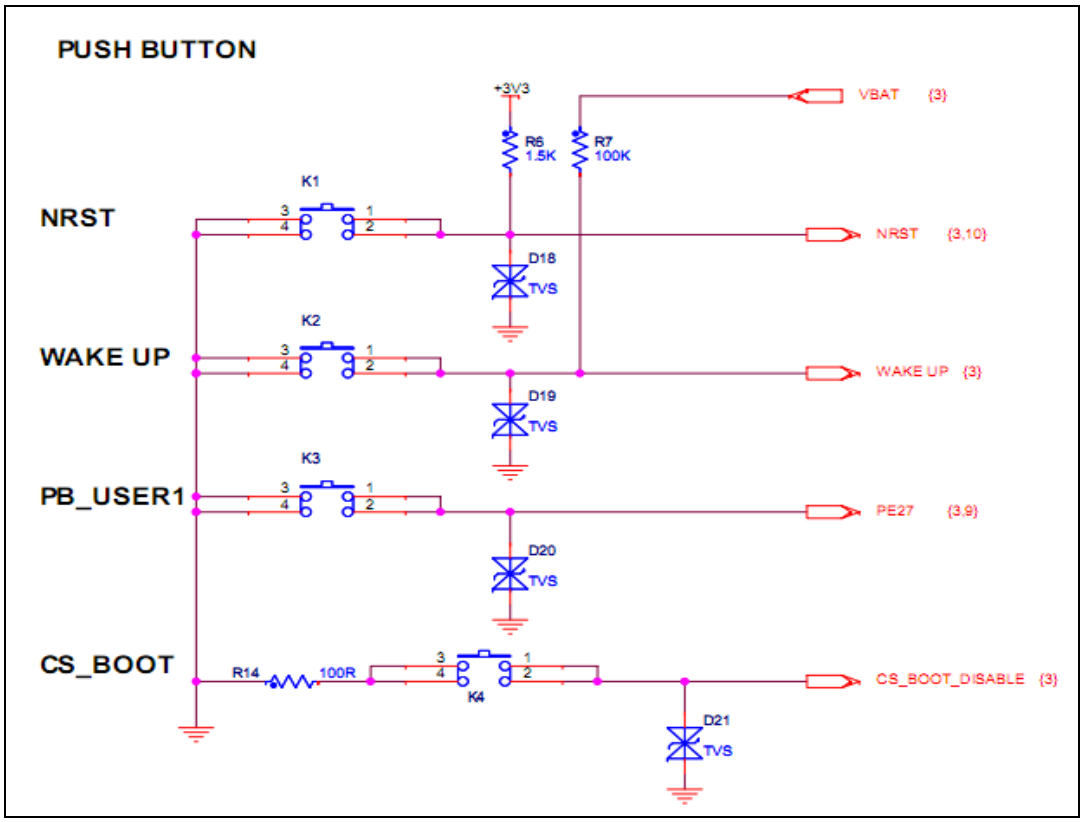


Figure 2-11

2.3.2 CAN

SAMA5D34 and SAMA5D35 have CAN receive/send interface. Its characteristics are as follows:

- Fully compatible with the “ISO 11898” standard
- High speed (up to 1Mbaud)
- Very low Electro Magnetic Emission (EME)

- Different receiver with wide common-mode range for high Electro Magnetic Immunity (EMI)
- An unpowered node does not disturb the bus lines
- Transmit Data (TxD) dominant time-out function
- Silent mode in which the transmitter is disabled
- Bus Pins protected against transients in an automotive environment
- Input levels compatible with 3.3V and 5V devices
- Thermally protected
- Short-circuit proof to battery and to ground
- At least 110 nodes can be connected

CAN receive/send circuit figure 2-14:

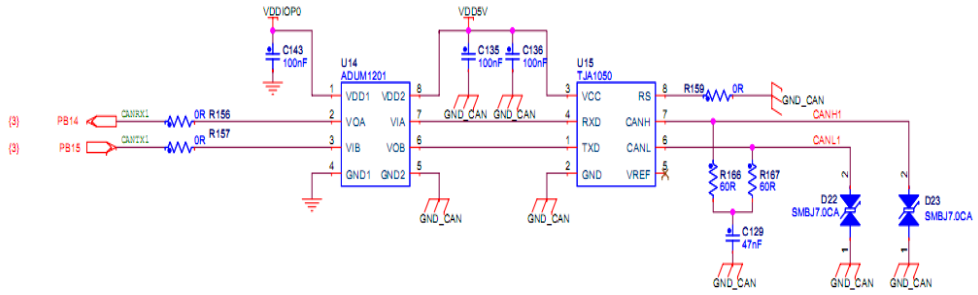


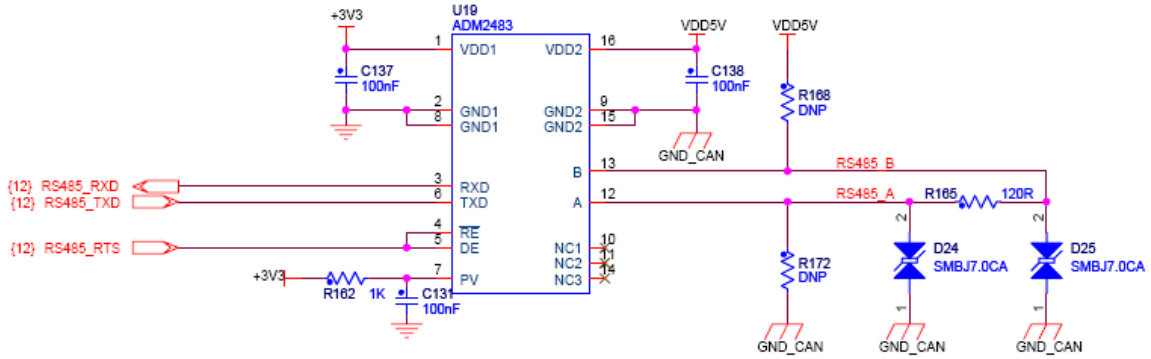
Figure 2-12

2.3.3 RS485

RS485 receive/send key feature:

- Electrical data isolation
- Complaince to ANSI TIA/EIA RS-485-A-1998 和 ISO 8482:1987(E)
- Transfer spped reach high as 500kbps
- low power consumption, current <=2.5mA
- compatible 5v and 3v power supply (VDD1)
- Bus device numbers <=256
- work temperature: -40℃ to +85℃

RS485 circuit digram 2-13



Digram 2-13

2.3.4 HSMCI Module

HSMCI Module is shown in figure 2-14:

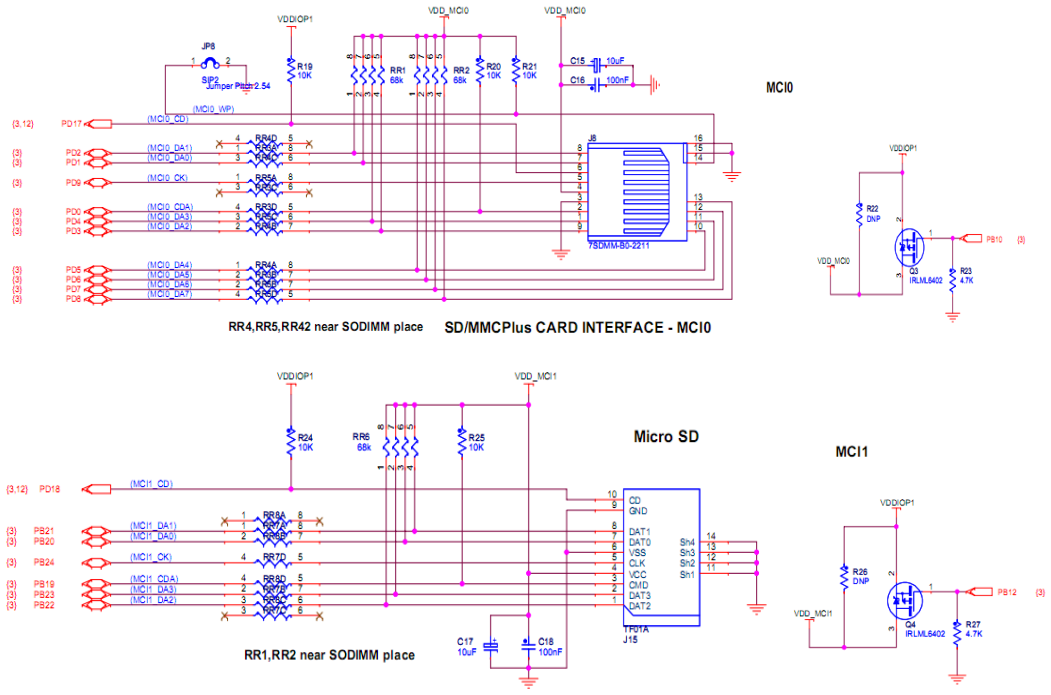


Figure 2-14

2.3.5 SODIMM Interface

(1) LCD Hardware interface circuit is shown in figure 2-15:

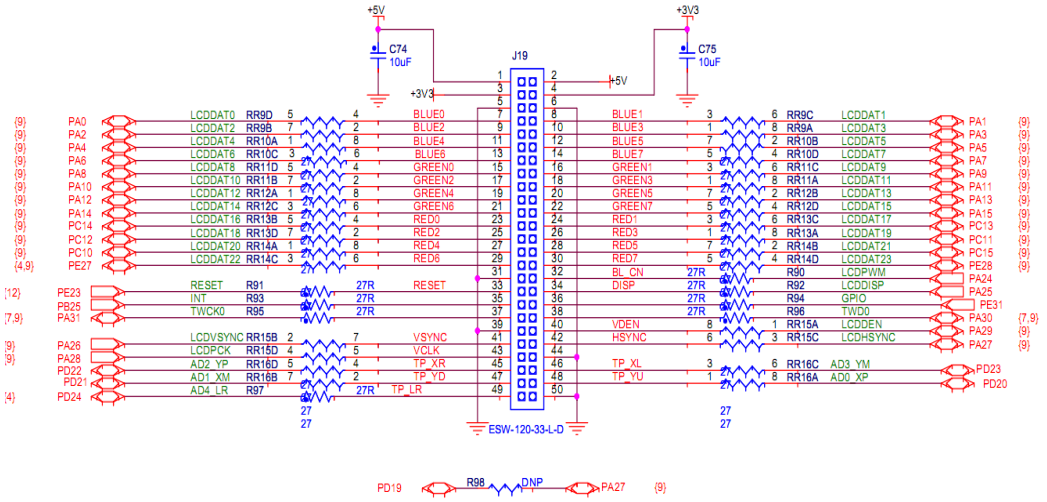


Figure 2-15

(2) TAG interface circuit is shown in figure 2-16:

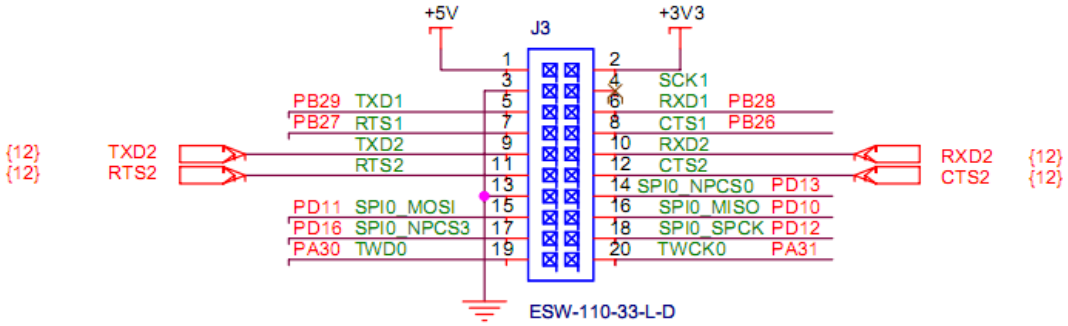
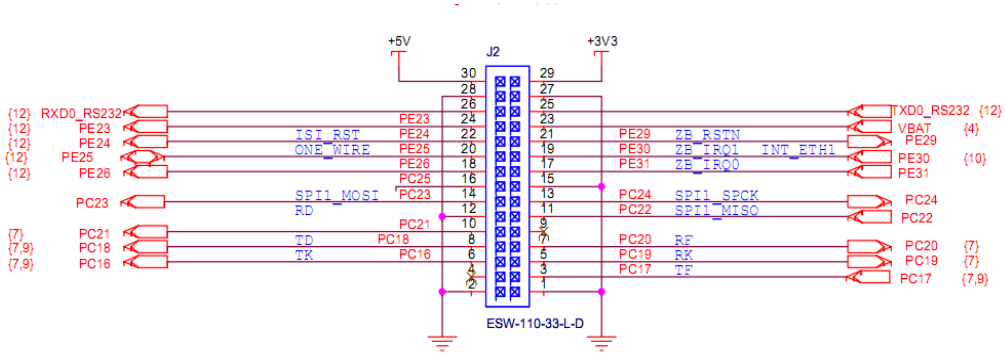


Figure 2-16

(3) User interface circuit is shown in figure 2-17:



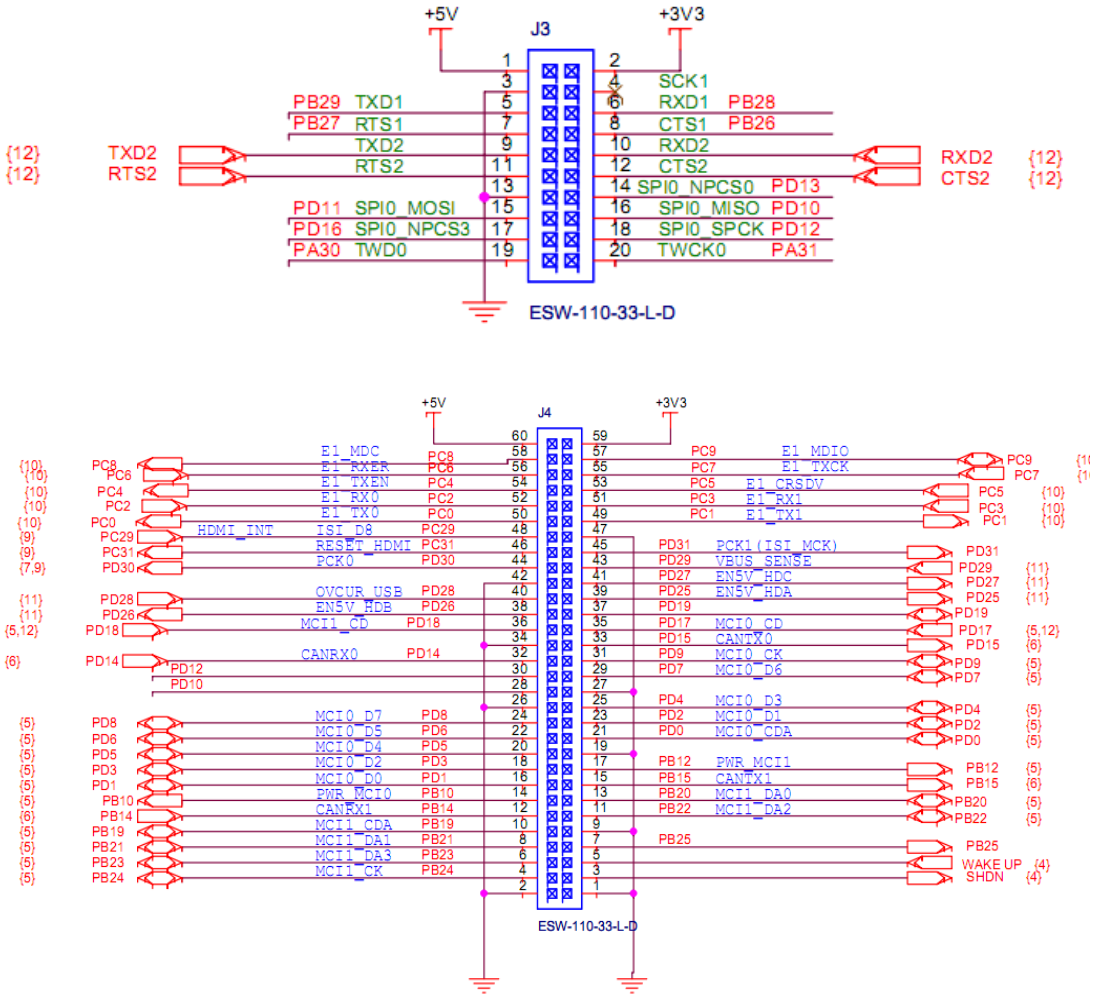


Figure 2-17

(5) ISI interface circuit is shown in figure 2-18:

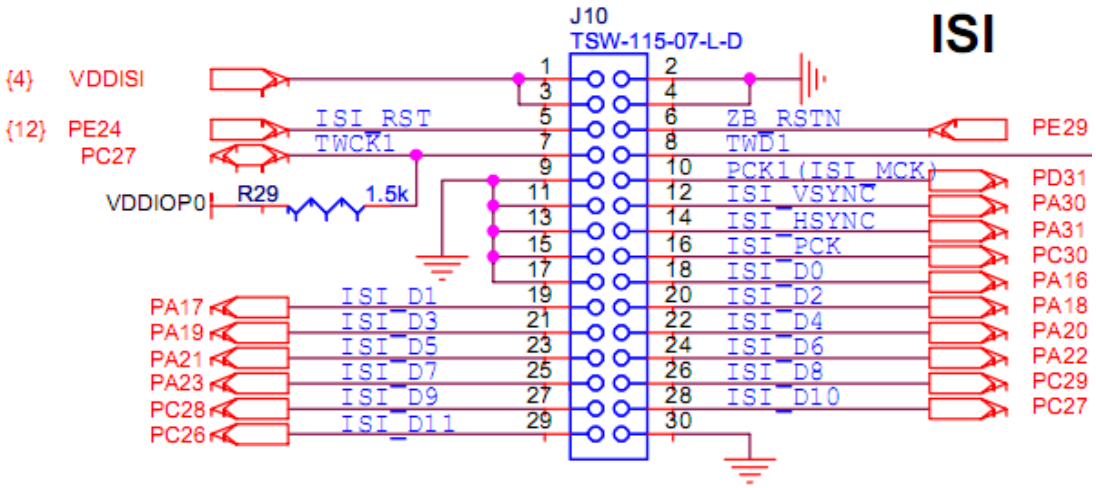


Figure 2-18

2.3.6 Smart DAA Module

SmartDAA Module is shown in figure 2-19:

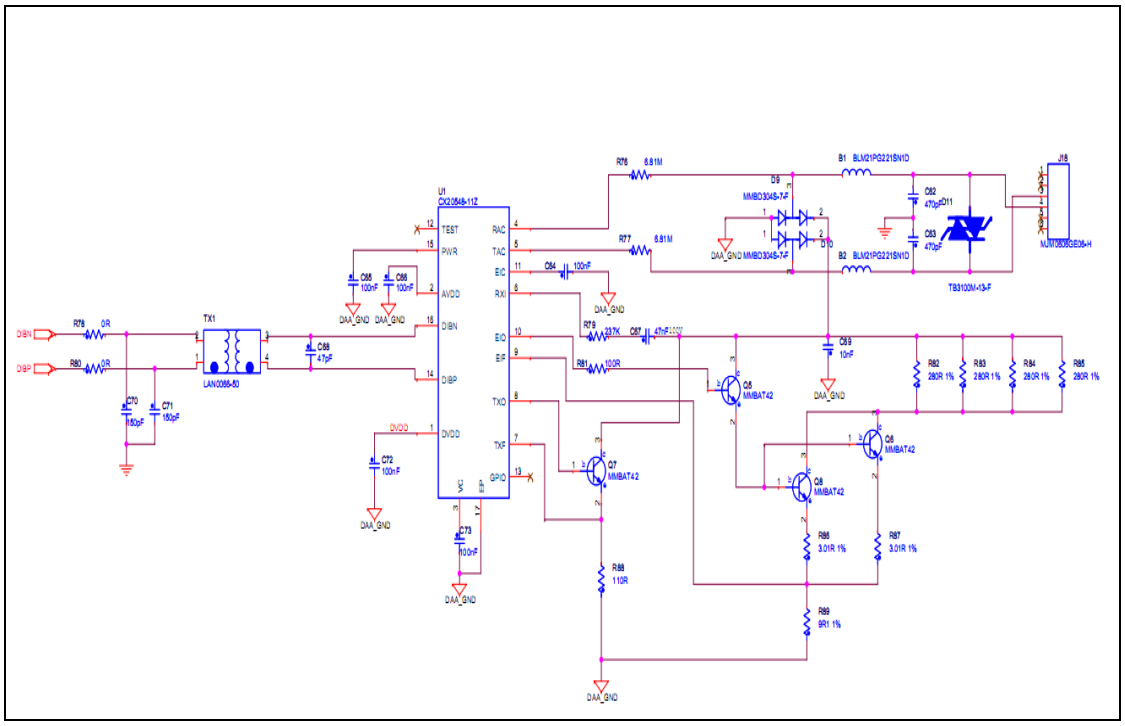


Figure 2-19

2.3.7 Audio Module

- Audio Performance
  - 91dB SNR ('A' weighted @ 48kHz) ADC
  - 96dB SNR ('A' weighted @ 48kHz) DAC
- ADC and DAC Sampling Frequency: 8kHz–96kHz
- 2 or 3-Wire MPU Serial Control Interface
- Programmable Audio Data interface Modes
  - I^2S, Left, Right Justified or DSP
  - 16/20/24/32 bit Word Lengths
  - Master or Slave Clocking Mode
- Stereo sound output and input
- The output and input volume control

- Highly Efficient Headphone Driver
- Playback only 3.0mW
- Analog Pass Through Power only 2.4mW
- 32-PIN QFN package(4 x 4mm, 0.4mm pitch)

Circuit is shown in figure 2-20:

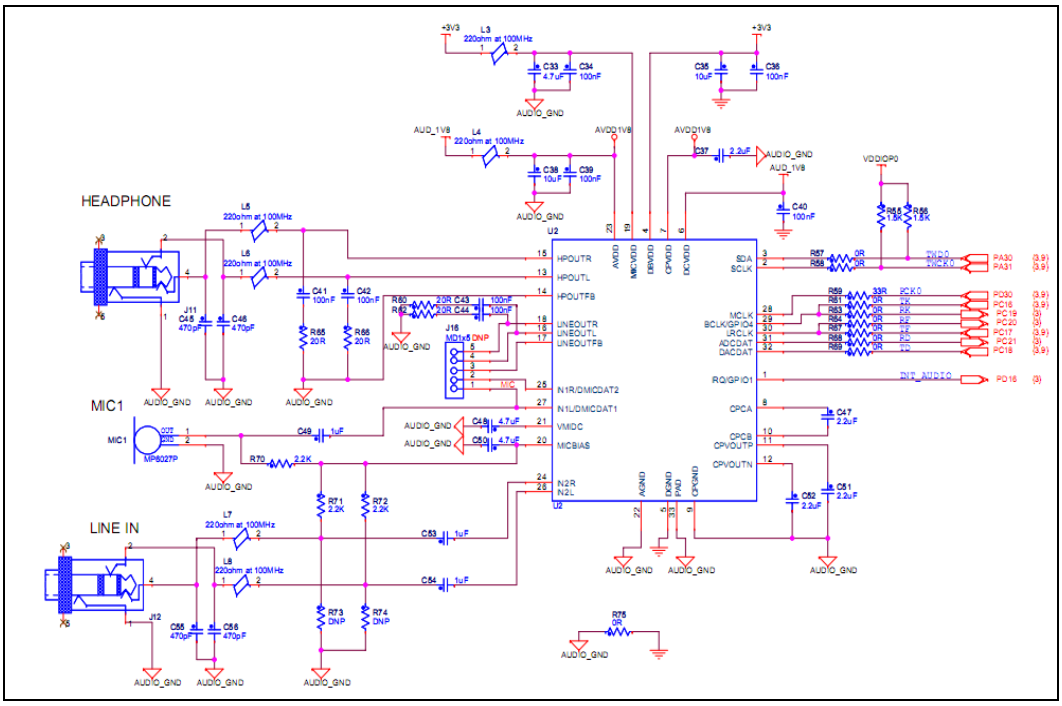


Figure 2-20

2.3.8 USB Module

(1) USB HOST mode and its characteristics are as follows:

- 110mΩ (5V Input) High-Side MOSFE Switch
- 500mA Continuous Load Current per Channel
- 110μA Typical On-State Supply Current
- 1μA Typical Off-State Supply Current
- Current-Limit/Short Circuit Protection
- Thermal Shutdown Protection under Overcurrent Condition
- Under voltage Lockout Ensures that Switch is off at Start Up
- Output can be Forced Higher than Input(Off-State)
- Open-Drain Fault Flag

- Slow Turn ON and Fast Turn OFF
- Enable Active-High or Active-Low

USB HOST Interface circuit is shown in figure 2-21:

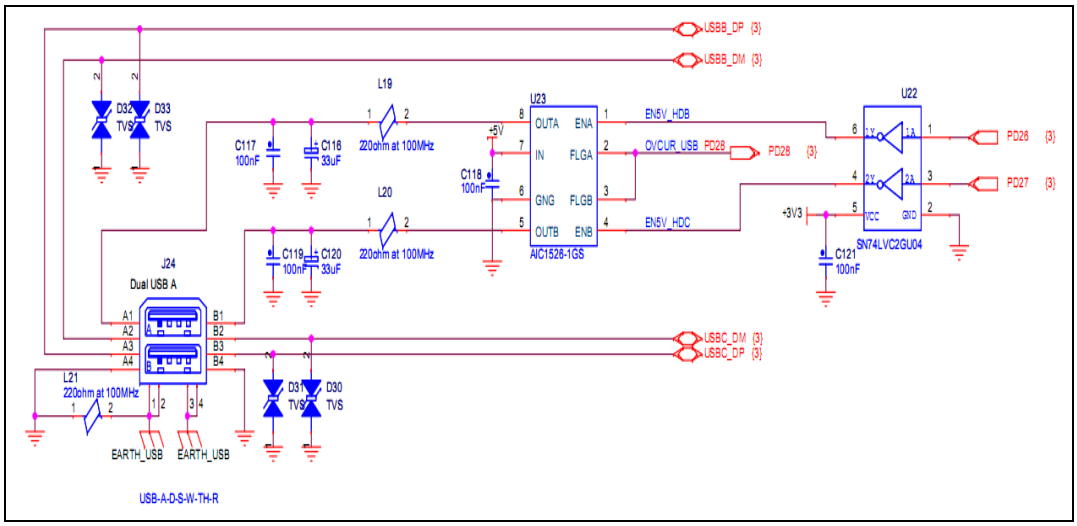


Figure 2-21

(2) mini USB Host / Device and its characteristics are as follows:

- 70-mΩ High-Side MOSFET
- 500 mA Continuous Current
- Thermal and short-Circuit Protection
- Accurate Current Limit(0.75A min, 1.25 A max)
- Operating Range:2.7V to 5.5V
- 0.6-ms Typical Rise Time
- Deglitched Fault Report
- Bidirectional Switch
- Ambient Temperature Range: –40°C to 85°C
- ESD Protection

Circuit is shown in figure 2-22:

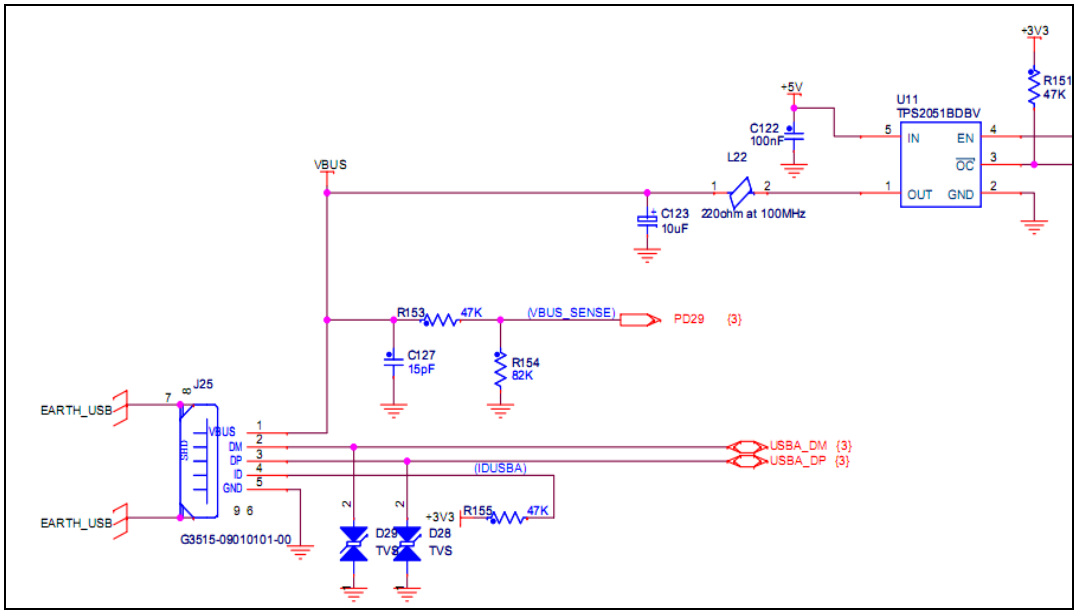


Figure 2-22

2.3.9 ETHERNET Module

ETHERNET's characteristics are as follows:

- Single-chip 10/100Mbps IEEE 802.3 compliant Ethernet Transceiver
- RGMII1.2 Interface support with 50MHz reference clock output to MAC, and option to input 50MHz reference clock
- Programmable interrupt output
- On-chip termination resistors for the differential pairs
- Baseline Wander Correction
- Power down and power saving modes
- 32-pin QFN package (5mmx 5mm)
- MDC/MDIO Management Interface for PHY register configuration
- Power supply:
  - VCC (core): 1.2 V
  - VCCQ (I/O): 3.3 V /2.5 V

ETH Module interface circuit is shown in figure 2-23:

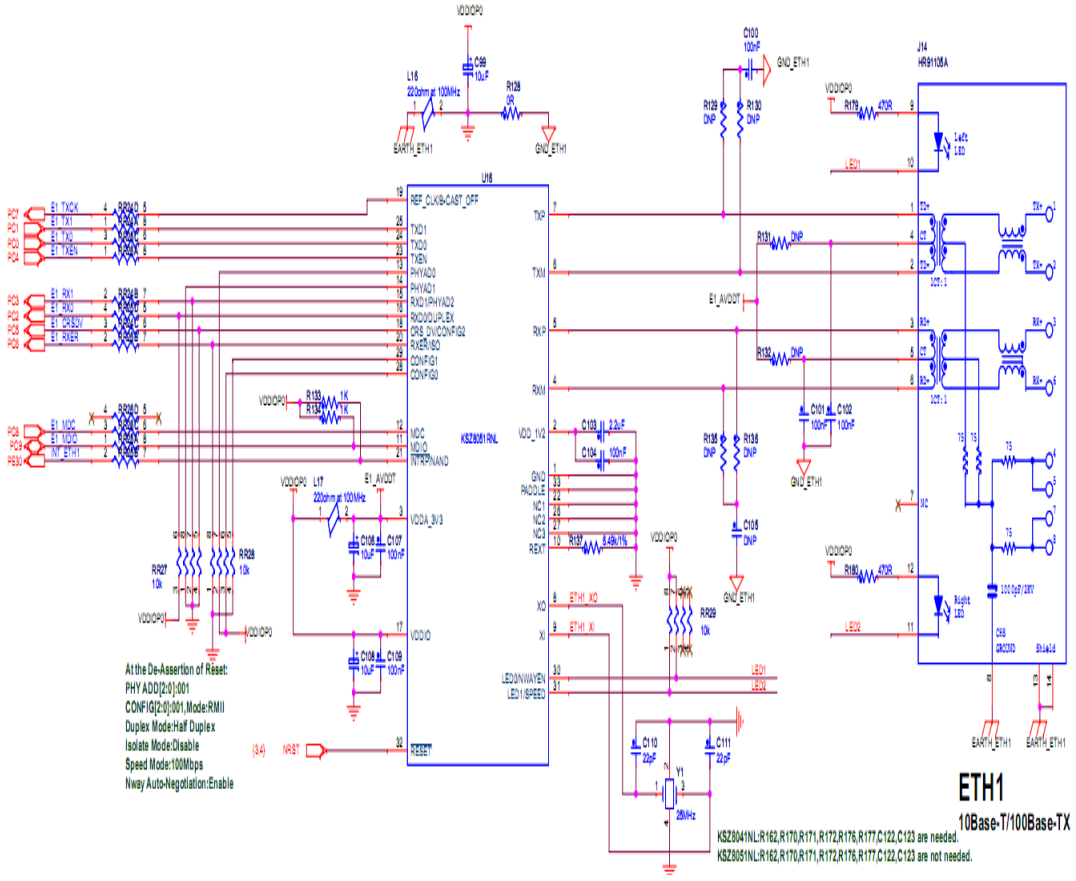


Figure 2-23

2.3.10 HDMI Module

HDMI key features

- Support HDMI1.3,SimplayHD,DVI1.0 protecol
- DVD-AUDIO support 4xIIS input
- Support 2-channel 192HZ or 8-channel 96HZ
- Programmable DDC control system design
- Flexible interrupt register
- 72-pin QFN package (10x10mm)
- Support ICE 60958 or ICE 61937
- Support MDO/MDIO management interface
  - VCC (core): 1.2 V
  - VCCQ (I/O): 3.3 V /2.5 V

HDMI Module interface circuit is shown in figure2-24:

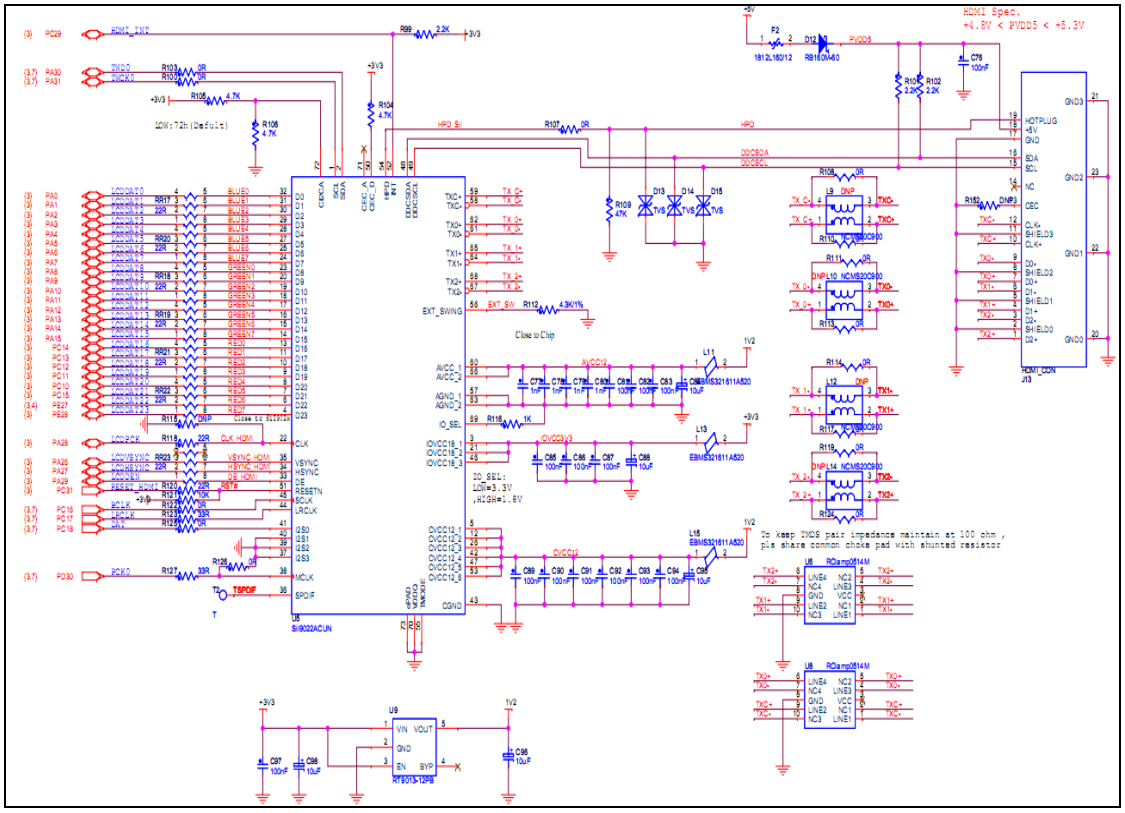


Figure 2-24

2.3.11 USART Module & DBUG

USART Module interface circuit is shown in figure 2-25:

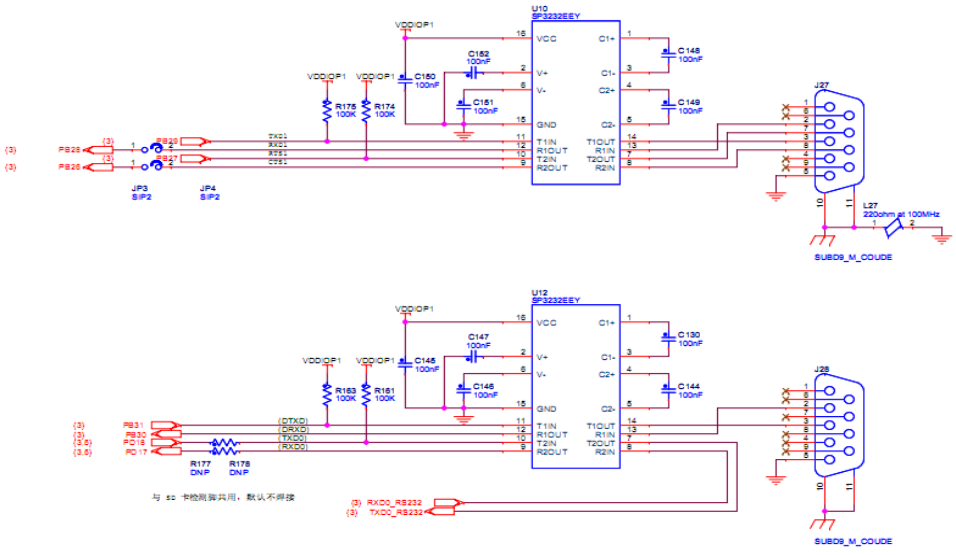


Figure 2-25

2.4 Jumper Setting

Table2-2 (Jumper Setting)

NO	Initial	Description	
		ON（Jumper close）	OFF（Jumper open）
JP1	OFF	Enable DataFlash（on CPU module）	Disable DataFlash（CPU module）
JP2	OFF	Boot from external Memory	Boot from ROM
JP3	ON	RXD1 signal connection	RXD1 signal close
JP4	ON	CTS1 signal connection	CTS1 signal close
JP5	ON	Power supply	Power supply by SHDN signal
JP6	ON	VBAT power supply	VBAT power close
JP7	1-2	(1-2)pin, VDDANA power supply 3V3	(2-3)pin，VDDANA power supply 3V
JP8	OFF	J8 SD card write protect	Disable J8 SD write protect
JP9	OFF	UART2 output at J26 as 485 signal	UART2 output at J3 as usart

Table 2-2

Chapter 3 MDK use

MDK will in the subsequent version update to add, the current version temporarily does not provide the MDK resources.

Chapter 4 Linux System Guide

4.1 Outline

This chapter describes how to run Linux system and embedded Linux applications, the process of drive development in MYD-SAMA5D3X development board. It includes the development environment to build, compile source code, image download and Linux application and driver example. The default startup is that Nand-Flash start the initial system. Product is Linux system at the factory and the Nand-Flash content distribution and some analysis are as bellows:

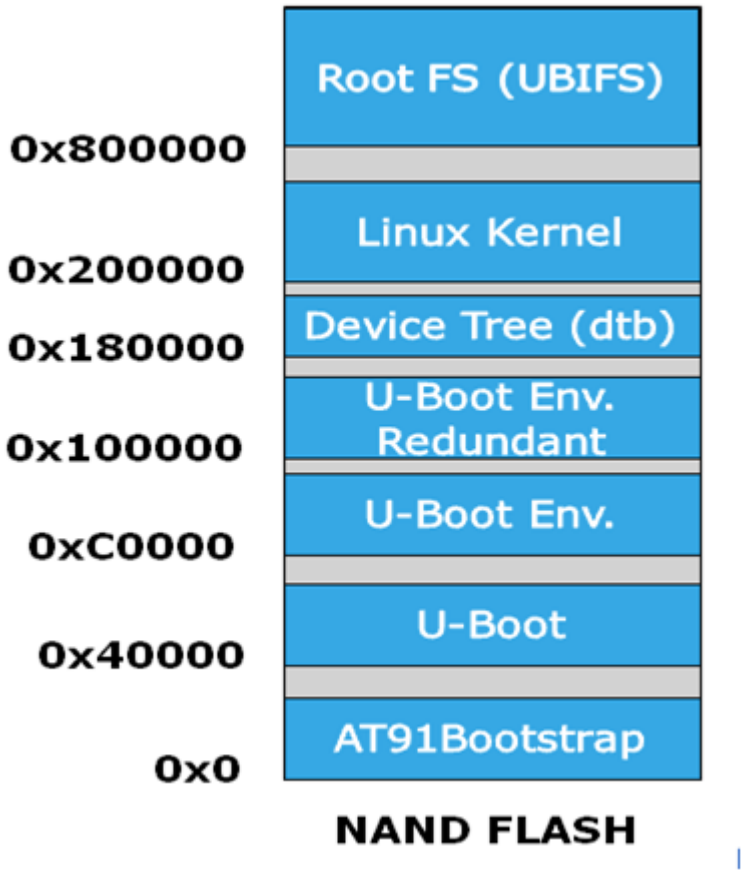


Figure 4-1

(1) Boot-Strap

After power on system, the first class boot program is copied automatically from NandFlash to internal SRAM and begins to implement by CPU. The main role is to

initialize CPU and external RAM and u-boot is copied from Nand-Flash to external RAM, and then jump to u-boot entry and start u-boot;

(2) u-boot

Secondary boot program, which is used for kernel image updates, load kernel and boot kernel starts;

(3) u-boot Env

Configure environment variables and provide u-boot running parameters, such as ip address, start a command, kernel boot parameters;

(4) sama5d3x.dtb

Describe hardware device tree, U-boot DTB in the start-up will Flash when the address of the file as the inlet parameters are passed to the kernel;

(5) Linux Kernel

Design Linux 3.6.9 kernel for MYD-SAMA5D3X;

(6) Root FS

The buildroot file system support QT graphical application.

4.2 Software Resources

Category	Name	Remark
Boot program	Boot Strap	First boot program, source code available
	u-boot	Secondary boot program, source code available
Linux kernel	Linux 3.6.9	Linux kernel only for MYD-SAMA5D3X hardware, source code available
Device Drivers	USB Host	USB Host driver supports the mode of OHCI and EHCI transmission, source code available
	USB Device	USB Device Driver (Gadget) , source code available
	Ethernet	Ethernet driver, source code available
	MMC / SD	MMC/SD Card driver, source code available
	Nand-Flash	Nand-Flash/Smart-Media driver, source code available

	TWI(I2C)	I2C driver, source code available
	SPI	SPI driver, source code available
	WM8904	WM8904 Audio driver, source code available
	LCD Controller	LCD driver, support 4.3 inch, Scalable 7 inch, source code available
	RTC	RTC clock driver, source code available
	Touch-Screen	4 -wire resistive touch screen driver, source code available
	PWM	PWM (pulse width modulation) driver, source code available
	USART	Serial port driver, source code available
	LED	LED driver, including GPIO LED PWM LED driver , source code available
System Files	QT	The buildroot file system support QT graphical application, source code available

Table 4-1

4.3 Start Linux System

4.3.1 Install Download Tool

Here use SAM-BA v2.12 provided by Atmel, it can find at DVD path: 03-Tools\SAM-BA\sam-ba_2.12.exe , specific installation method, please refer to the document -Tools\SAM-BA\sam-ba install.pdf.

Install pack sam-ba_2.12_patch4.exe after sam-ba_2.12.exe.

4.3.2 Connect Board to PC

⑴ Connect Board to PC (Order cannot be reversed) ,Specific steps are as follows:

① Development board switch to 5 v (power off-position)

② Through the mini-USB line connected to the PC and development board

③ Disonnect the CPU module jumper JP1, JP2 backboard disconnect, hold CS_BOOT button At the same time development board switches to USB_5V. IF first time the PC opportunities prompted to install driver development board, Selected at this time SAM - BA installation directory under the relevant position can be installed as shown in figure 4-2:

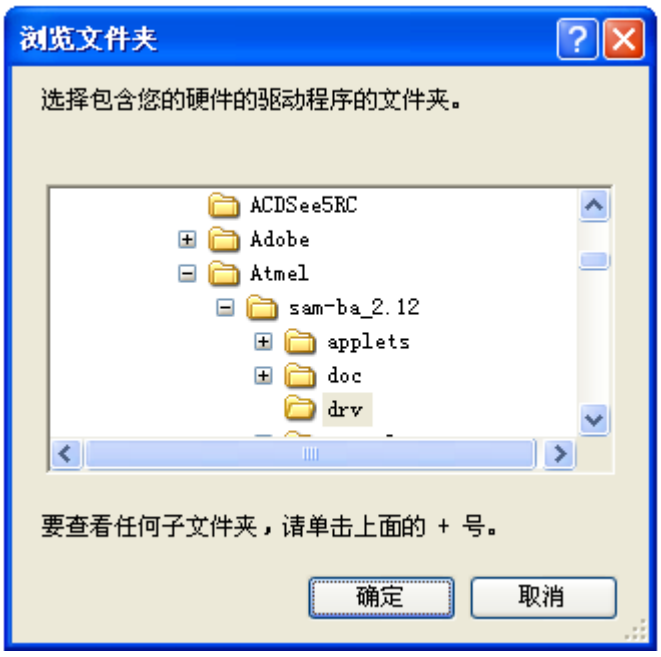


Figure 4-2

⑵ If there is figure 4-3 in " computer->properties->Management-> device manager-> port", which shows board driver has been installed (According to the actual situation, here for COM4).

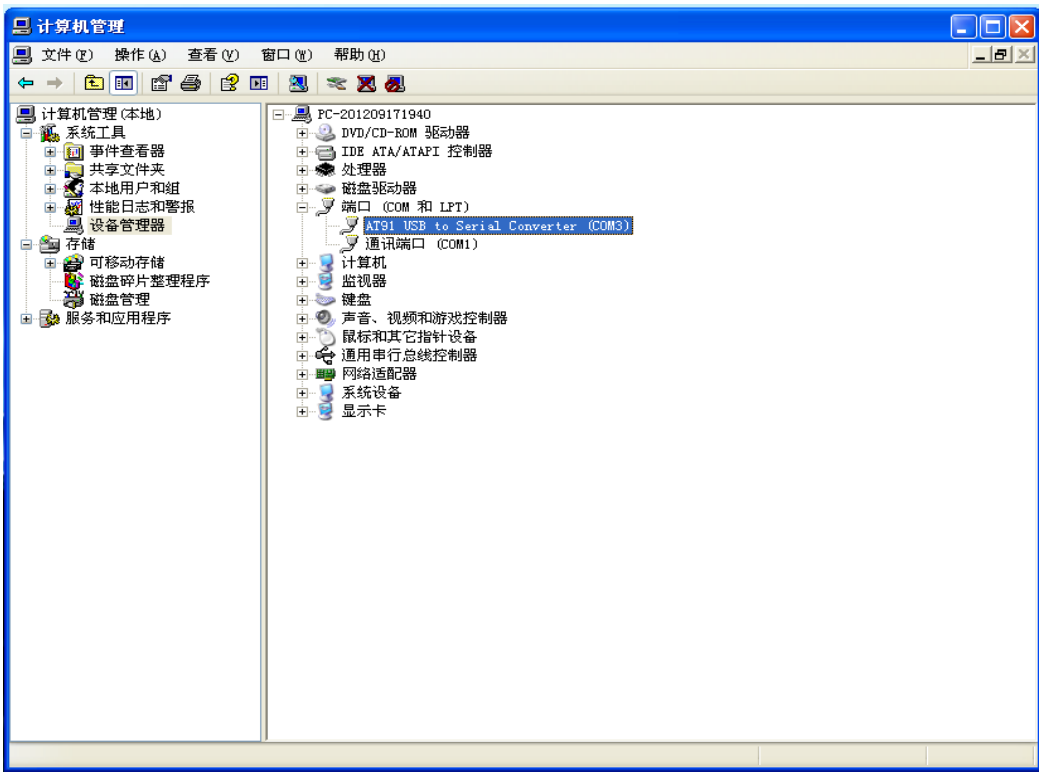


Figure 4-3

(3) Connect J28 to PC by serial cable, set up SerialTerminal: Baud rate115200, data bit 8, no parity, stop bit 1, no rts/cts. COM port number is set by actual situation.

4.3.3 Automatic Download

Note: please pull out SD card before download, otherwise an error may occur when programing.

After complete chapter 4.3.1 and 4.3.2 the board will be as a USB device by PC, open CD-ROM directory: MYIR_SAMA5D3X\02-Images\Linux-image, double-click [sama5d3xek_demo_linux_nandflash]. Then SAM- BA will download Linux image automatically to board. Entire download process takes ab- out three minutes. When pop logfile.log file automatically, reset board, there will be Linux start information.

Linux use, please refer to chapter 4.7.

4.3.4 Manual Download

Note: please pull out SD card before download, otherwise an error may occur

Use SAM-BA to download Linux manually

(1) Complete chapter 4.3.1 and 4.3.2, open SAM-BA to set corresponding parameters. Connection is \the USB-serial\COMXX (XX is each computer's COM port, choose it by actual situation, here selected COM8) board select at91sama5d3x-ek. And then click “Connect”, specific settings and connected results are shown in figure 4-4, 4-5:

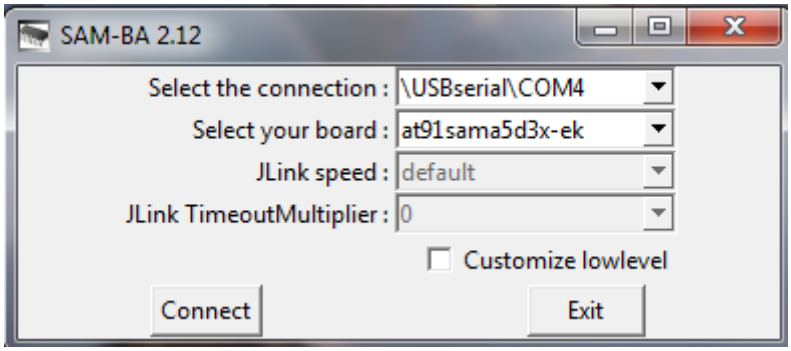


Figure 4-4

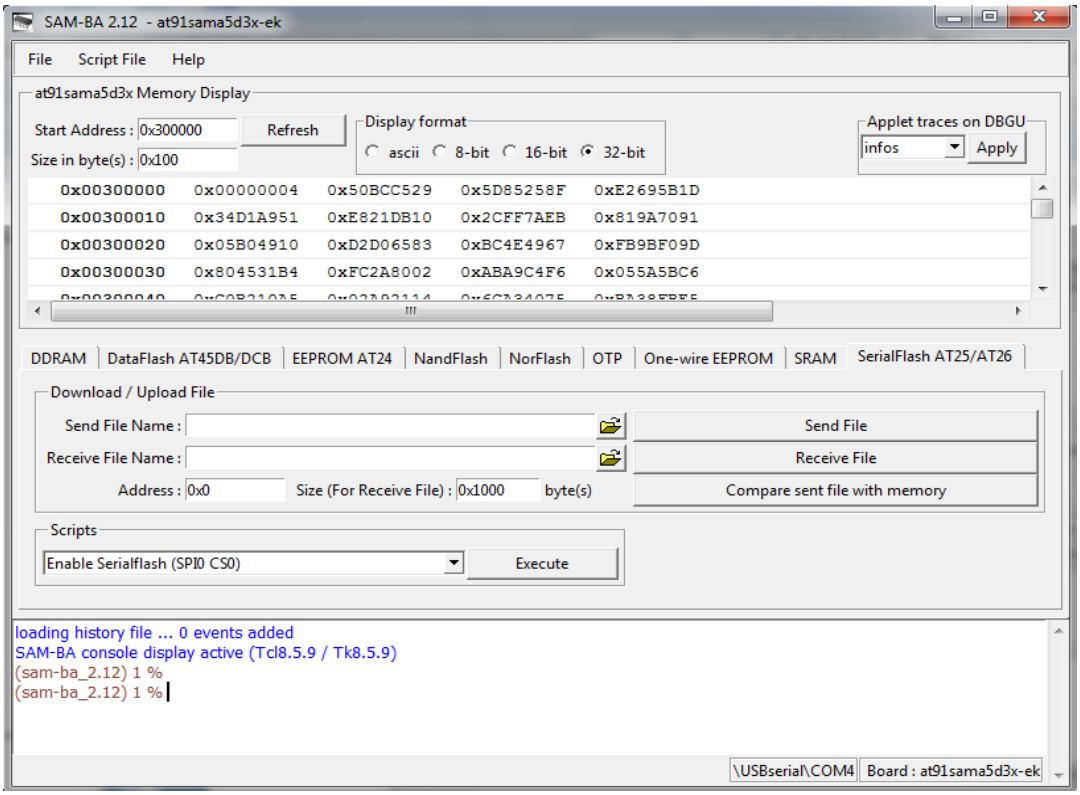


Figure 4-5

(2) Click Nand-Flash tab and execute Enable Nand-Flash Erase All, Enable OS PMECC parameters in Scripts tab (select an action and click next to "Execute" execution). When Execute Enable OS PMECC, Pop - up dialog box, click OK to use the default settings, the specific operation is as follows in figure 4-6:

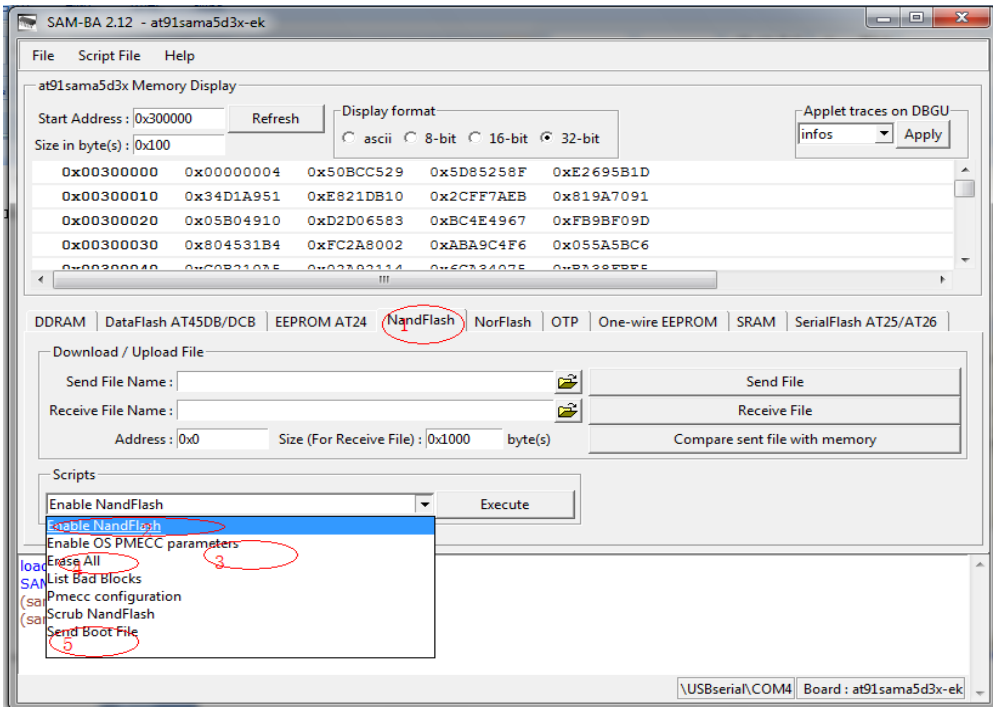


Figure 4-6

① Select "**Enable Nand-Flash**" in Scripts tab, and then click "**Execute**" to Enable Nand-Flash. Refer to figure 4-7:

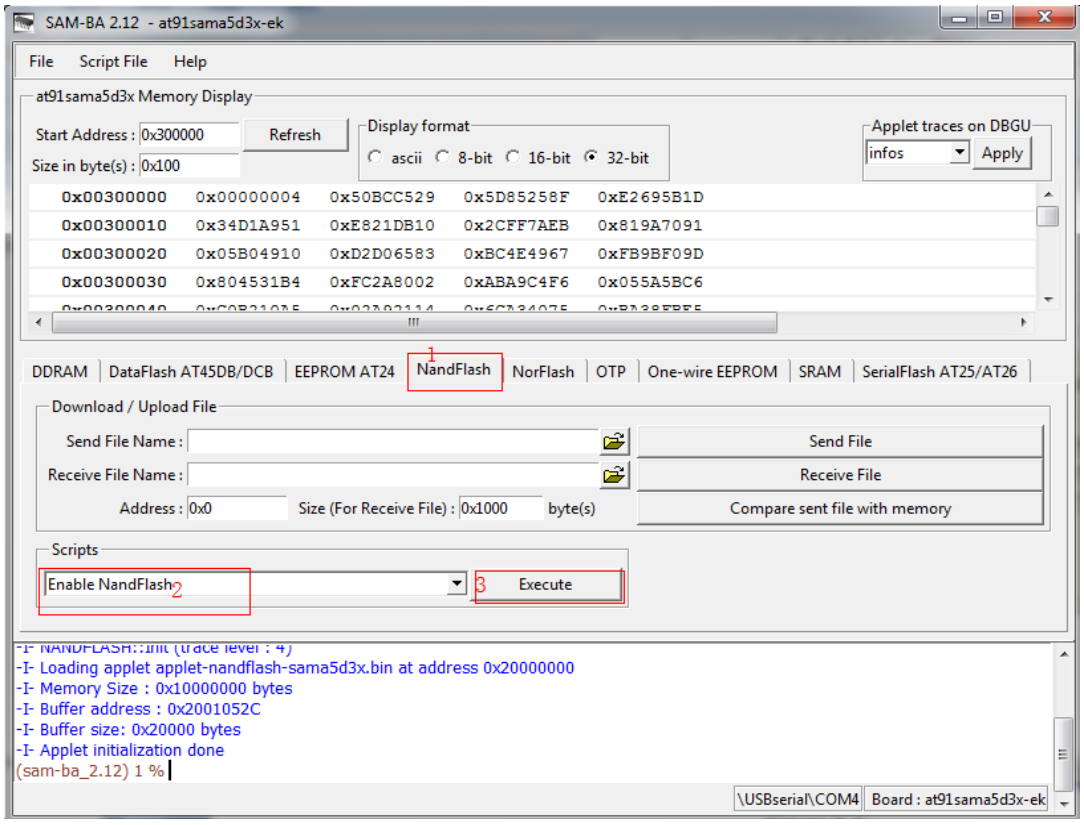


Figure 4-7

② Select **"Enable OS PMECC parameters"** in Scripts tab and then click "**Execute**", S ettings refer to figure 4-8 (Note :do not select Trimffs):

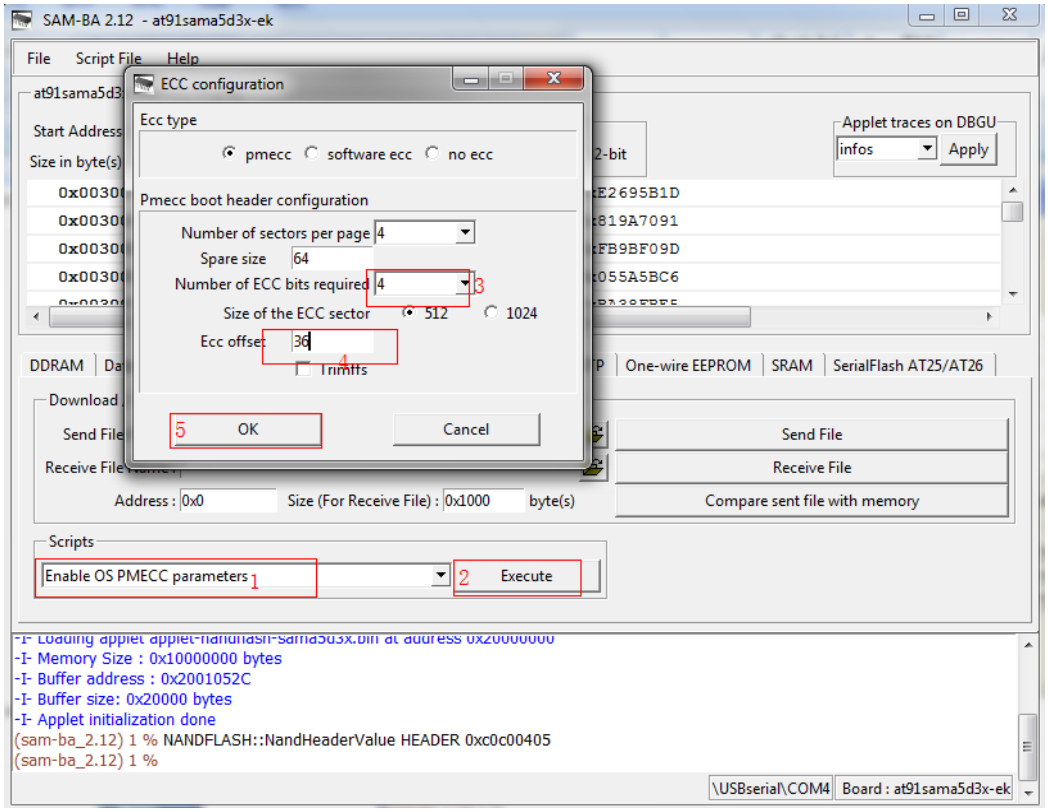


Figure 4-8

③ Select "**Erase All**" in Scripts tab, then click "**Execute**", format Nand-Flash. Refer to figure 4-9:

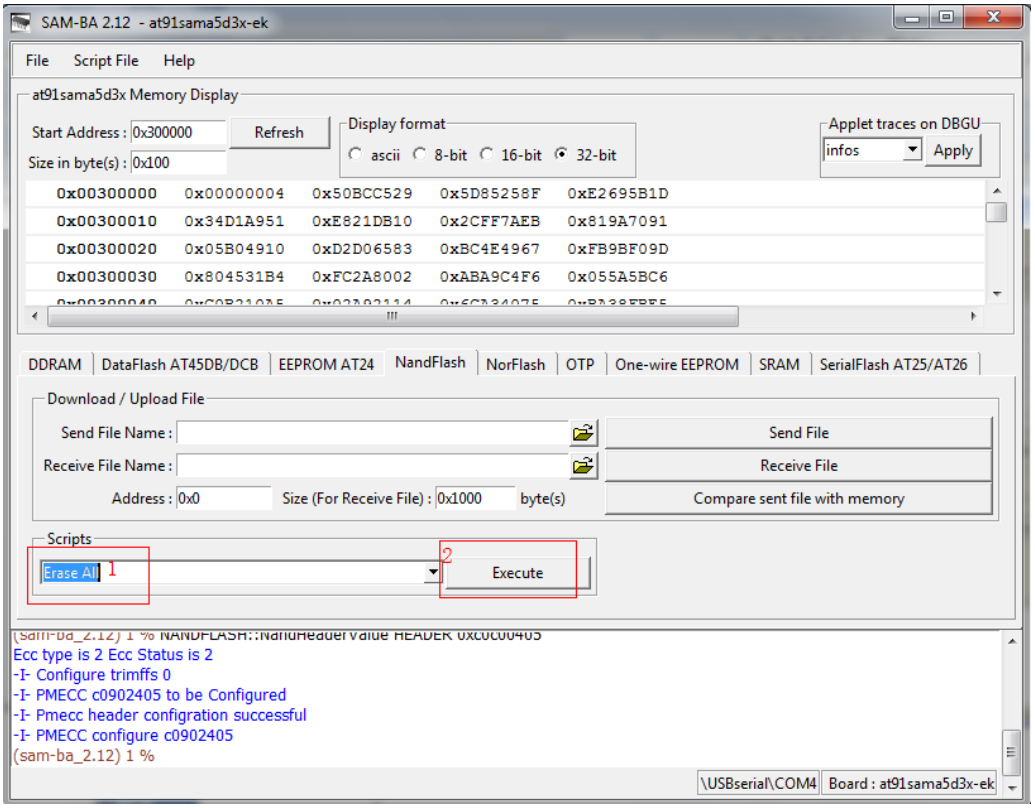


Figure 4-9

④ Download [boot. bin] Refer to figure 4-10 and 4-11:

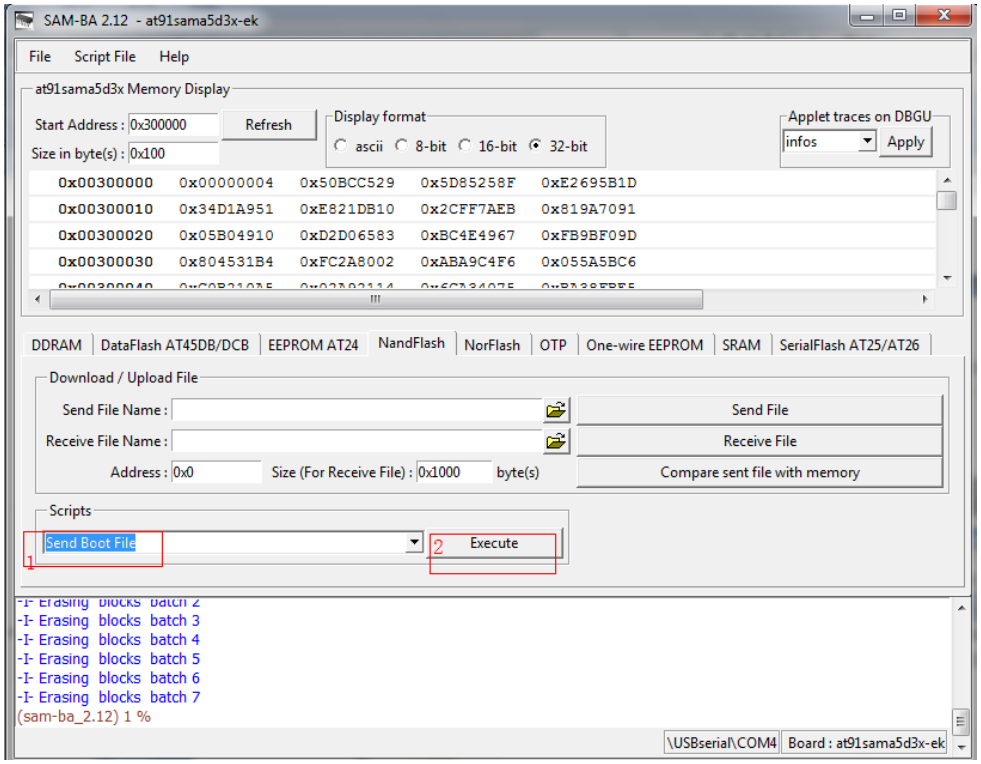


Figure 4-10

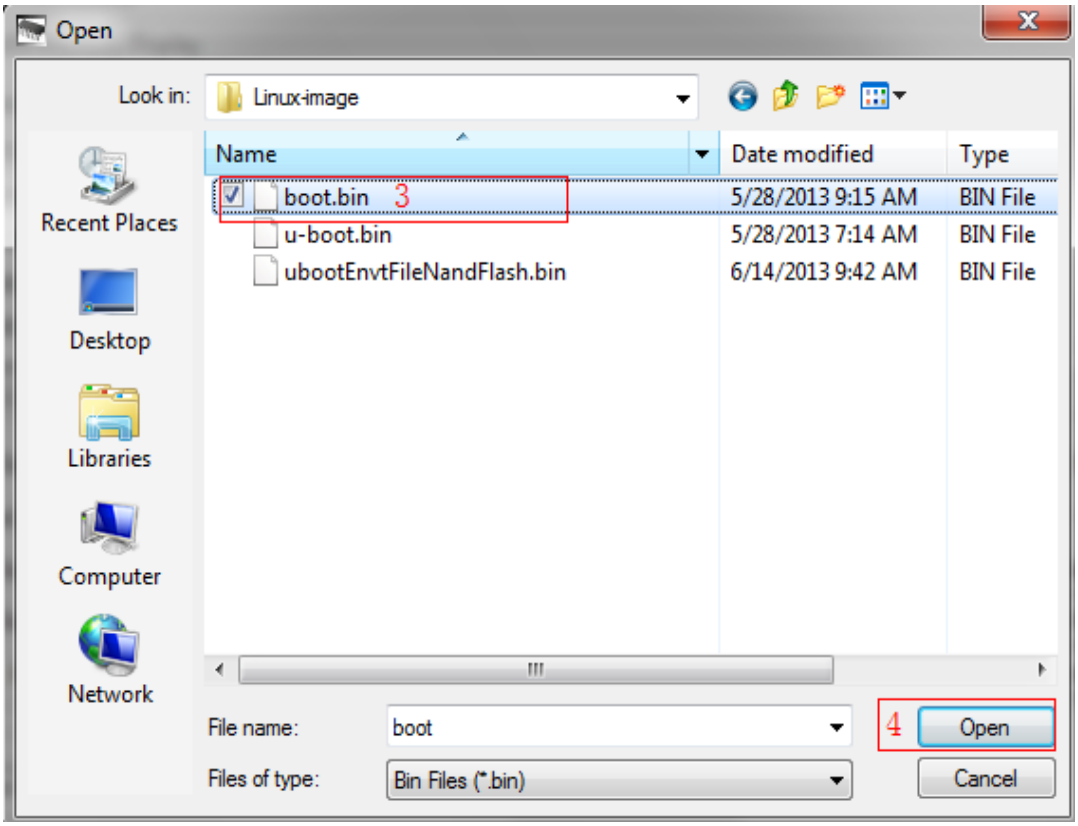


Figure 4-11

⑤ Download [u-boot. bin] to 0x40000. Refer to figure 4-12:

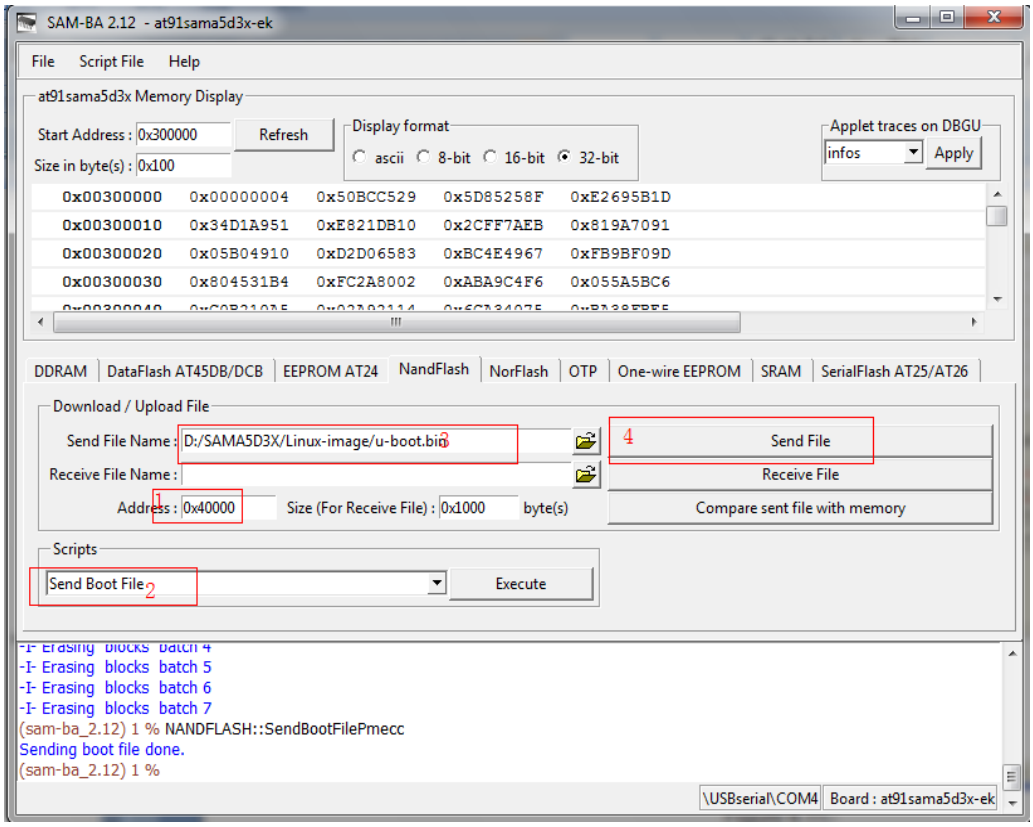


Figure 4-12

⑥ Download [ubootEnvtFileNand-Flash. bin] (this file generated by sam-ba when programing) to 0x000C 0000. Refer to figure 4-13:

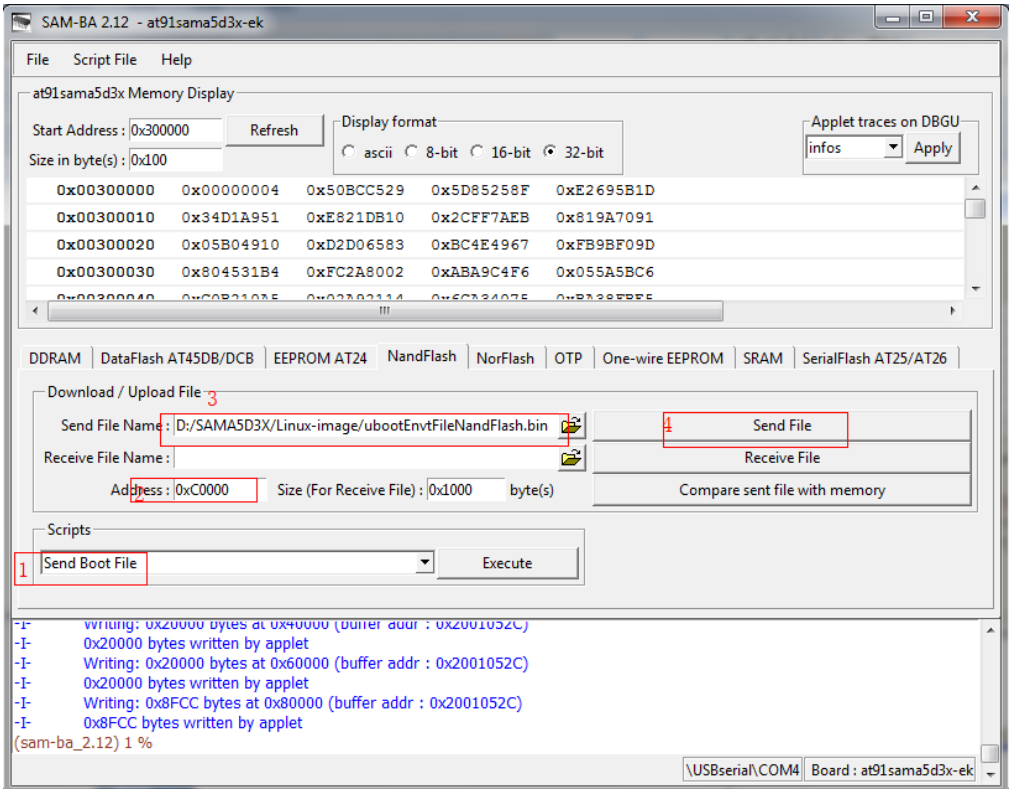


Figure 4-13

⑦ Download [sama5d3xek.dtb] to 0x0018 0000. Refer to figure 4-14:

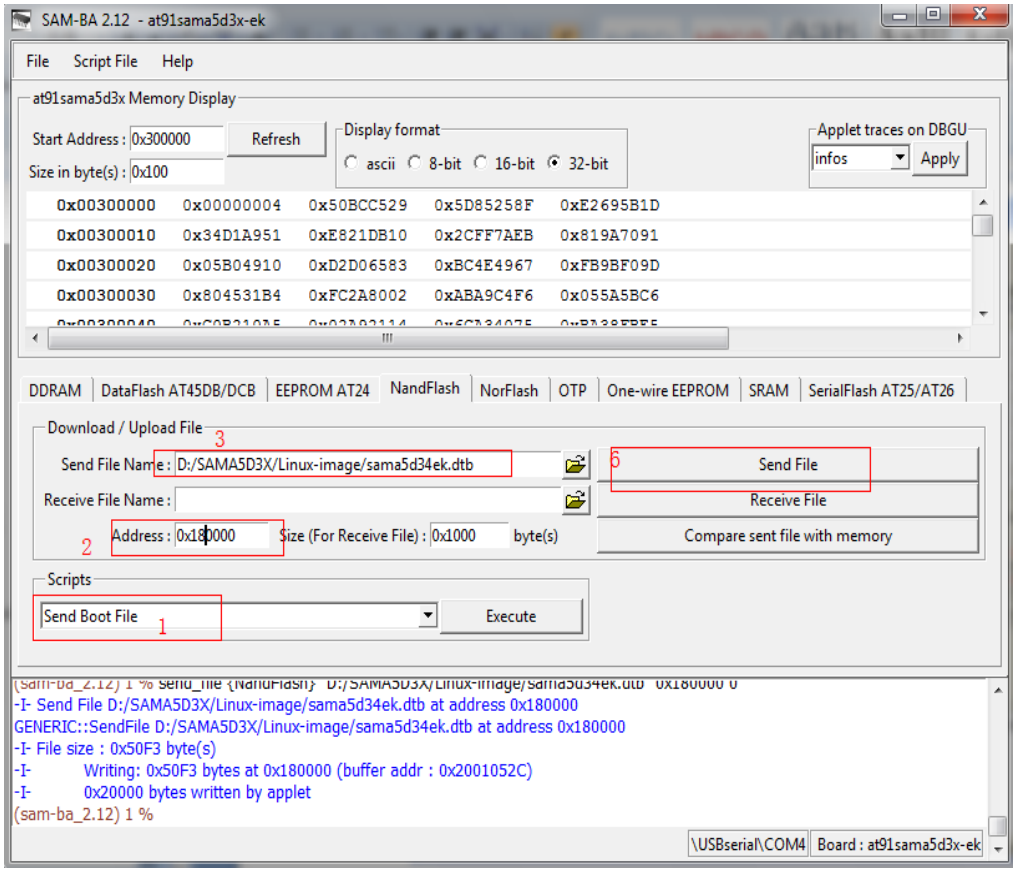


Figure 4-14

⑧ Download linux kernel [uImage] to 0x0020 0000. Refer to figure 4-15 figure 4-16:

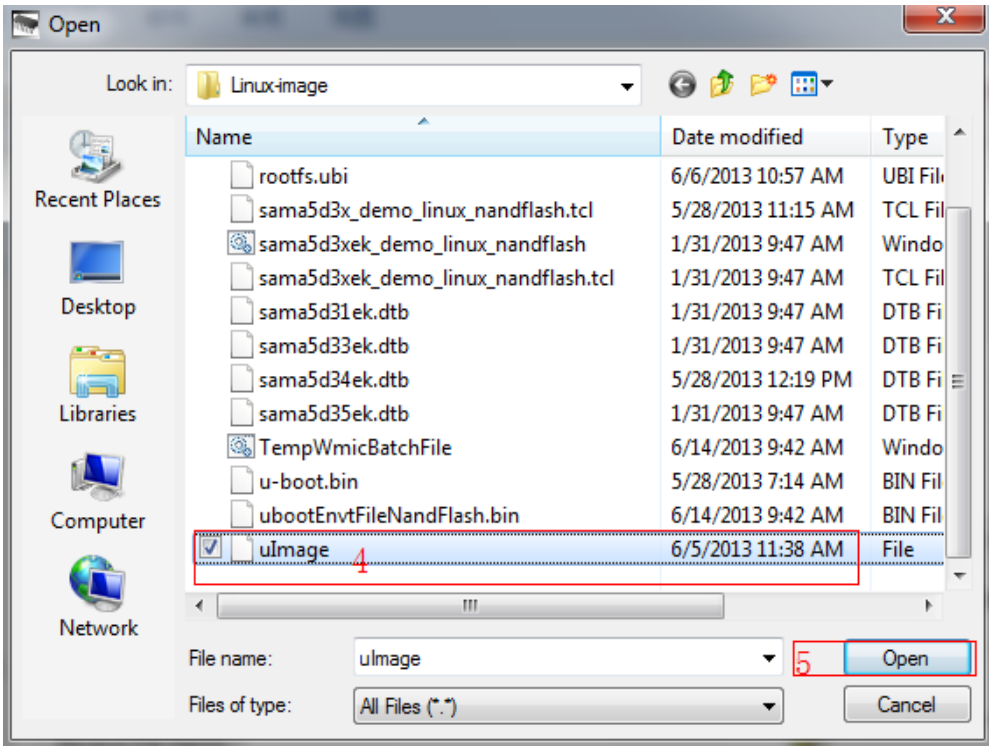


Figure 4-15

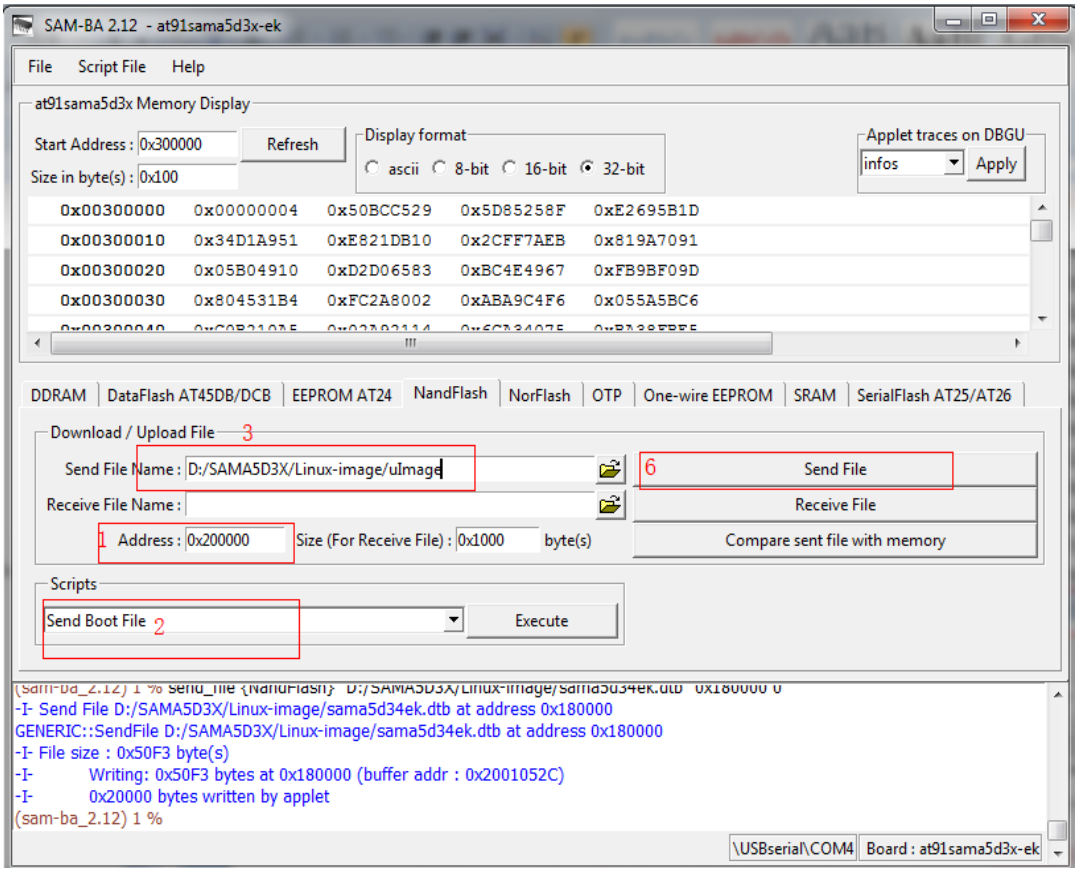


Figure 4-16

⑨ Download [rootfs. ubi] to address: 0x80000. Refer to figure 4-17(Note: Select Enable OS PMECC parameters, and click “Execute” to select Trimffst.):

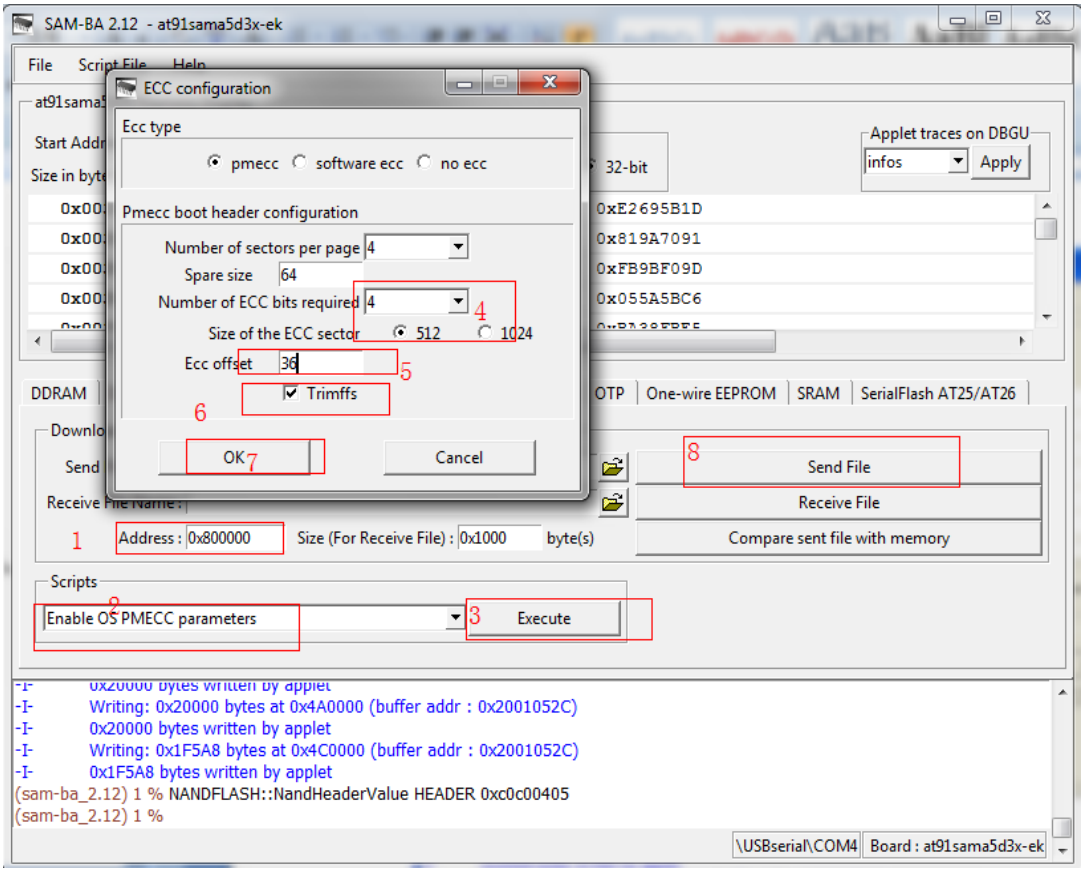


Figure 4-17

(3) Finally, restart board to boot Linux system normally. Linux use, please refer to chapter 4.7.

4.4 Linux Development Environment Structure

The contents of this chapter, please refer to **"description VirtualBox 's Linux - based development environment to build pdf"**.

4.5 Installation and Compile

4.5.1 Create a Working Directory

Linux in the working directory and CD of the copy of the folder on the/opt/linux (the user can also customize your own working directory)

```
# mkdir /opt/linux
# cd /opt/linux
```

```
# cp -r /media/cdrom/05-Linux_Source/* ./
```

4.5.2 Install Cross Compiler Tools

Decompress cross compiler tool chain

```
# sudo apt-get install build-essential git libncurses5-dev u-boot-tools
# sudo apt-get install flex bison texinfo zip unzip zlib1g-dev gettext
# sudo apt-get install gperf libsdl-dev libesd0-dev libwxgtk2.6-dev
# cp /media/cdrom/03-Tools/Cross_compiler/gcc-linaro-arm-linux-gnueabihf
-4.7-2013.04-20130415_linux.tar.bz2 ./
# sudo tar xf gcc-linaro-arm-linux-gnueabihf-4.7-2013.04-20130415_linux.tar.bz2 -C
/opt/
# export ARCH=arm
#export
CROSS_COMPILE=/opt/gcc-linaro-arm-linux-gnueabihf-4.7-2013.04-20130415_linux/bin
/arm-linux-gnueabihf-
```

4.5.3 Install AT91Bootstrap Source and Compile

(1) Install

```
# cd /opt/linux/Bootstrap
# tar xvjf at91bootstrap.tar.bz2
# cd at91bootstrap
```

(2) Compile

```
# make mrproper
# make at91sama5d3xeknf_uboot_defconfig
# make
```

At91SAMA5D3Xek-Nand-Flashboot-3.1.bin in binary directory is AT91Bootstrap file. Rename it to boot.bin for automatic download. AT91Bootstrap is a boot loader for ATMEL chip, which initialize necessary hardware (GPIO Clock, SDRAM, etc.), then copy uboot to SDRAM to run.

4.5.4 Install uboot Source and Compile

(1) Install

```
# cd /opt/linux/U-boot
# tar xvjf u-boot-at91.tar.bz2
# cd u-boot-at91
```

(2) Compile

```
# make distclean
# make sama5d3xeknf_512MB_config
# make
```

U-boot.bin is U-boot file that we download in u-boot-at91 directory

4.5.5 Install and Compile Linux kernel Source Code

(1) Install

```
# cd /opt/linux/Linux-at91
# tar xvjf linux-at91.tar.bz2
# cd linux-at91
```

(2) Compile

```
# export ARCH=arm
# make sama5d3_defconfig
# make
# sudo apt-get install uboot-mkimage
# make uImage
# make dtbs
```

After compile kernel, uImage file in arch /arm/boot/ directory is Linux image files that we need. sama5d31ek.dtb, sama5d33ek.dtb, sama5d34ek.dtb, sama5d35ek.dtb is Linux device tree image file what we need.

4.5.6 Install and Compile Linux File System

(1) Install

```
# cd /opt/linux/Buildroot-at91
# tar xvjf buildroot-at91.tar.bz2
# cd buildroot-at91
```

(2) Compile

```
# sudo apt-get install flex bison texinfo unzip gettext
# sudo apt-get install g++ xz-utils
# cp buildroot.config .config
# make
```

It takes long time to compile kernel, rootfs.ubi in arch/arm/boot/ directory is Linux file system that we need.

4.6 Make Linux File System

rootfs.ubi can be made a simple formulation and revision in system file provided by CD. Here, take hellomyir: for an example, add an application to file system root directory, show the detailed steps of making file system.

4.6.1 Write Demo Program hellomyir

Firstly, write a simple program hellomyir:

(1) Creat and compile hellomyir.c

```
# vi hellomyir.c
```

Enter the following in the hellomyir.c, save and exit:

```
# include <stdio.h>
int main(int argc, char *argv[])
{
    int i;
    printf("========== Hello Myir ==========\n");
    printf("argc: %d\n", argc);
    for(i = 0; i < argc; i++)
```

```
        {
            printf("argv[%d]: %s\n", i, argv[i]);
        }
        return 0;
    }
```

(2) Compile hello myir.c

Add cross-compiler tools path to PATH:

```
# export PATH=$PATH: /opt/
gcc-linaro-arm-linux-gnueabihf-4.7-2013.04-20130415_linux/bin/arm-linux-gnueabihf-
```

Use the cross compiler tool to compile:

```
# arm-none-linux-gnueabi-gcc -static -o hellomyir hellomyir.c
```

“hello myir” application is generated

4.6.2 Mount UBIFS File System

UBIFS is a new flash file system by nokia engineers under the help of Szeged University, which is considered the next generation of JFFS2 files system. UBIFS file system specifically for the large-capacity FLASH embedded mobile devices, mounting UBIFS file system must have mtd interface, while Ordinary PC usually has no mtd manage disk. So here needs nandsim simulator, simulate mtd device with a RAM space, then mount UBIFS file system. The concrete steps are as follows:

(1) Load UBIFS drive mtd driver

Enter the following command by turn:

```
# sudo apt-get install mtd-utils
# sudo modprobe nandsim first_id _byte=0x2c second_id_byte=0xda
  third_id_byte=0x90 fourth_id_byte=0x95
```

It needs to pass a few parameters to load nandsim: first_id_byte, second_id_byte, third_id_byte and fourth_id_byte are ID for NAND-FLASH of the simulated target. MYD-SAMA5D3X use NAND-FLASH of Samsung K9F2G08U0B. Four ID bytes can be found in datasheet: 0x2c，0xda，0x90，0x95

If executed successfully, there will be mtd0 and mtd0r0 devices in /dev/directory:

```
# ls /dev/mtd*
/dev/mtd0     /dev/mtd0ro
```

(2) Erase mtd0 partition, the operation is as follows:

```
# sudo flash_eraseall /dev/mtd0
Erasing 128 Kibyte @ ffe0000 -- 100 % complete
```

(3) Load rootfs.ubi to new mtd0 partition. Here used dd command to load, as follows:

```
# sudo dd if= rootfs.ubi    of=/dev/mtd0
94976+0 records in
94976+0 records out
48627712 bytes (49 MB) copied, 0.73911s, 65.8MB/s
```

(4) Mount UBIFS system file

After completing the above step, load ubi modules and attach to mtd0, mount UBIFS file system like mounting ordinary mtd device.

Load ubi modules and attach to mtd0 equipment:

```
# sudo   modprobe   ubi   mtd=0,2048
```

Create a new mount point:

```
# mkdir fsmount
```

Mount it by the following command:

```
# sudo   mount   -t   ubifs   ubi0_0   fsmount/
# ls fsmount/
bin   boot   dev   etc   home   lib   media   mnt   proc   sbin   sys   tmp   usr   var
```

Now that mount UBIFS file system is successful.

4.6.3 Modify UBIFS System Files

After mount UBIFS file system successfully, it can modify file contents, such as add, delete and modify files. It should add compiled demo program helloworld to system root directory. The operation is as follows:

```
# sudo cp hellomyir fsmount/
```

```
# sync
# ls fsmount
bin boot dev etc hellomyir home lib media  mnt  proc  sbin  sys  tmp  usr  var
```

4.6.4 Regenerate UBIFS System File

After modification, it needs to regenerate file system by mkfs.ubifs tool, Enter the following command to generate a new UBIFS file system:

```
# sudo  mkfs.ubifs  -r  fsmount/  -m  2048  -e  126976  -c  2024  \
   -o  ubifs.img
```

mkfs.ubifs Parameter Description:

-r Establish the system file directory

-m Minimum I/O transfer unit size

-e Logical size of erase block

-c The largest number of erase logic blocks

-o Specify the output file

View smallest I/O transfer unit size and logical erase block size by the following command:

```
# ubinfo /dev/ubi0
ubi0
Volumes count:                           1
Logical eraseblock size:                 126976 bytes, 124.0 KiB
Total amount of logical eraseblocks:     2048 (260046848 bytes, 248.0 MiB)
Amount of available logical eraseblocks: 0 (0 bytes)
Maximum count of volumes                 128
Count of bad physical eraseblocks:       0
Count of reserved physical eraseblocks:  20
Current maximum erase counter value:     1
Minimum input/output unit size:          2048 bytes
Character device major/minor:            250:0
```

```
Present volumes:                    0
```

View erase block number by using the following command:

```
# ubinfo /dev/ubi0_0
Volume ID:   0 (on ubi0)
Type:        dynamic
Alignment:   1
Size:        2024 LEBs (256999424 bytes, 245.1 MiB)
State:       OK
Name:        rootfs
Character device major/minor: 250:1
```

Then use ubinize tool to generate fsimage.ubi file. Firstly, it needs to create configuration files of ubinize.cfg:

```
# vi ubinize.cfg
```

Enter the following, save and exit:

```
[ubifs]
mode=ubi
image=ubifs.img
vol_id=0
vol_size=64MiB
vol_type=dynamic
vol_name=rootfs
vol_flags=autoresize
vol_alignment=1
```

Enter the following command to generate final fsimage.ubi file:

```
# sudo ubinize -m 2048 -p 128KiB -o fsimage.ubi ubinize.cfg
```

Parameter description of ubinize:

-m The size of minimum input/output byte flash unit

-p The erase block size of FLASH physical

-o output file

Here is different mkfs.ubifs parameter, - p parameter represents the physical erase

block size. UBI work in MTD layer, so it needs the MTD parameters, namely physical parameters. UBIFS work in UBI, so it needs the UBI parameters, namely the logic parameter. Now, UBI image has been saved in the ubi.img, which not only contains UBIFS information, but also contains UBI information.

After the completion, generated fsimage.ubi file can use the method described in chapter 4.3.4 to download to 0x800000

Reset board and input root to login, there is added hellomyir file in the root directory:

```
buildroot login: root
# cd /
# ls
bin          etc          lib          proc         tmp
boot         hellomyir    media        sbin         usr
dev          home         mnt          sys          var
```

Run helloworld, as follows:

```
# ./hellomyir
========== Hello Myir ==========
argc: 1
argv[0]: ./hellomyir
```

4.7 Linux Use

After a Linux system running, can be operated by touch screen, can also through the terminal serial ports, here is how to through the terminal Linux operating, such as U disk, SD card mount, ETH test and how to play music and so on.

4.7.1 Touch Screen Calibration

Entering system will run the screen calibration procedure automatically. Recalibrate the touch screen by the following steps if it requires:

(1) Open HyperTerminal (baud rate: 115200 Data bits: 8, Parity: None Stop bits: 1, data flow control: none). After start Linux, log in as root command:

```
buildroot login: root
```

(2) Run the calibration procedure and click the five corresponding calibration points on the LCD screen. The calibration can be carried out:

```
# ts_calibrate
xres = 480, yres = 272
Took 12 samples...
Top left : X =  767 Y =  178
Took 14 samples...
Top right : X =  793 Y =  947
Took 13 samples...
Bot right : X =  309 Y =  934
Took 12 samples...
Bot left : X =  308 Y =  169
Took 10 samples...
Center : X =  557 Y =  556
-29.638855 -0.011581 0.495638
331.720245 -0.364475 0.006420
Calibration constants: -1942412 -758 32482 21739618 -23886 420 65536
```

(3) After calibration is complete, it needs to restart system calibration to take effect. The operation is as follows:

```
# sync
# reboot
```

4.7.2 U disk Use

(1) Enter Linux by terminal, U disk is inserted to any of a USB host port, and you can see the following information in the HyperTerminal:

```
usb 1-3: new high-speed USB device number 2 using atmel-ehci
usb 1-3: New USB device found, idVendor=13fe, idProduct=3600
usb 1-3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
```

```
usb 1-3: Product: DT 101 G2
usb 1-3: Manufacturer: KINGSTON
usb 1-3: SerialNumber: C1690905
scsi0 : usb-storage 1-3:1.0
scsi 0:0:0:0: Direct-Access     KINGSTON DT 101 G2          8.07 PQ: 0 ANSI: 4
sd 0:0:0:0: [sda] 16076800 512-byte logical blocks: (8.23 GB/7.66 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
 sda: sda1
sd 0:0:0:0: [sda] Attached SCSI removable disk
```

(2) Entering the following command to mount inserted U disk and view U disk contents.

```
# mount -t vfat /dev/sda1 /mnt/
# ls /mnt/usb/
??                          MYD-S5PV210
??                          MYD-SAMA5D3X-Dev-Res
??-??-???-??-??             Michael Jackson - Beat It.mp3
?????                       Dangerous.mp3
BOOTEX.LOG                  Qt_Arm.rar
DSC04807.JPG                qt????
FOUND.000
```

(3) Unplug U disk directly when the use is completed, system will uninstall automatically

4.7.3 SD Card Use

(1) When Micro SD card is inserted, HyperTerminal displays SD card information:

```
mmc0: host does not support reading read-only switch. assuming write-enable.
mmc1: new high SD card at address aaaa
```

```
mmcblk0: mmc1:aaaa SU02G 1.84 GiB
mmcblk0: p1
```

(2) Mount and view SD card:

```
# mount -t vfat /dev/mmcblk0p1 /media/sdcard
# ls /media/sdcard/
360                    LOST.DIR              duba
360Download            Music                 iReader
Android                PicStore              mm_sec.dat
Audio                  QQ                    qqservicec.log
Backup                 QQBrowser             scan.txt
BaiduMapSdk            Qzone                 tencent
BlackBerry             RMS                   tmp
DCIM                   Sounds                wandoujia
Download               audio_play_list.txt
GT-S3600C.samsung      bluetooth
```

(3) Pull out SD card directly, system will uninstall it automatically.

4.7.4 Play MP3 Music

Before playing music, connect headphones or stereo to J11 U disk storages an mp3 music and is inserted into USB interface.

play music in U disk by mplayer command in terminal:

```
# mount –t vfat /dev/sda1 /mnt/
# ls /mnt
# madplay /mnt/Dangerous.mp3
```

At this point, it can hear music from headphones. Terminal prints information as shown below, enter Ctrl+C to end playing music:

```
MPEG Audio Decoder 0.15.2 (beta) - Copyright (C) 2000-2004 Robert Leslie et al.
```

4.7.5 Network Port Test

(1) The development board connected to a PC using crossover Ethernet cable, through the super terminal, for a development board configuration is not the IP address of the current local area network (LAN) other equipment takes up. in this case use address: 192.168.0.2:

```
# ifconfig  eth0  192.168.0.2  up
IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
# macb f0028000.ethernet: eth0: link up (1000/Full)
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
```

(2) Test board to PC network by ping command (here host IP: 192.168.0.3).

In PC ping development board: # ping 192.168.0.10 refer to Figure-4.18:

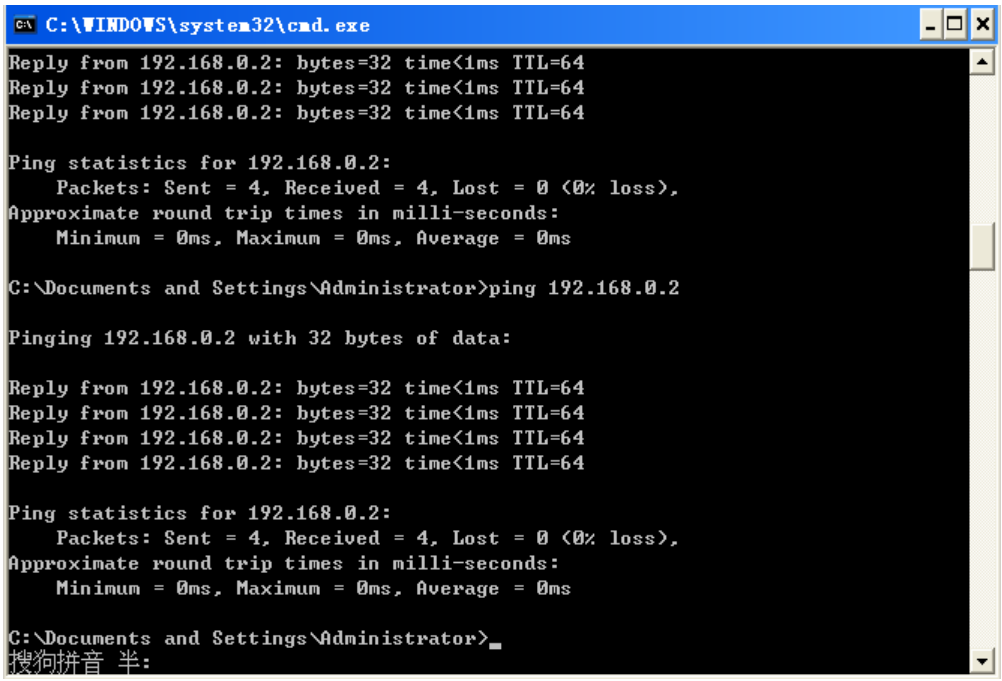


Figure-4.18

Ping the host in the development board:

```
# ping 192.168.0.3
PING 192.168.0.3 (192.168.0.3): 56 data bytes
64 bytes from 192.168.0.3: seq=0 ttl=128 time=0.801 ms
64 bytes from 192.168.0.3: seq=1 ttl=128 time=0.377 ms
64 bytes from 192.168.0.3: seq=2 ttl=128 time=0.357 ms
64 bytes from 192.168.0.3: seq=3 ttl=128 time=0.359 ms
```

```
64 bytes from 192.168.0.3: seq=4 ttl=128 time=0.447 ms
```

4.7.6 Telnet Test

(1) Configure IP address

```
# ifconfig eth0 192.168.1.2 up
```

(2) Configure Gateway

Test connection with the gateway, as follows:

```
# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=16.455 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=2.391 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=3.924 ms
64 bytes from 192.168.1.1: seq=3 ttl=64 time=2.254 ms
64 bytes from 192.168.1.1: seq=4 ttl=64 time=2.005 ms
```

(3) Set 192.168.0.1 as the default gateway:

```
# route add default gw 192.168.0.1
```

Test connection with 202.112.17.137:

```
# ping 202.112.17.137
PING 202.112.17.137 (202.112.17.137): 56 data bytes
64 bytes from 202.112.17.137: seq=0 ttl=54 time=32.202 ms
64 bytes from 202.112.17.137: seq=1 ttl=54 time=24.457 ms
64 bytes from 202.112.17.137: seq=2 ttl=54 time=22.918 ms
64 bytes from 202.112.17.137: seq=3 ttl=54 time=23.178 ms
64 bytes from 202.112.17.137: seq=4 ttl=54 time=29.341 ms
```

(4) Configure DNS server

View current DNS server address by ipconfig/all command, machine DNS is 202.103.24.68. Set target board's DNS (depend on the circumstances):

```
# echo "nameserver 202.103.24.68" | tee /etc/resolv.conf
```

Ping www.baidu.com to test extranet access:

(5) Use Telnet access to the BBS

```
# telnet    bbs.scut.edu.cn

华南木棉 BBS  最近  (1,10,15)  分钟平均负荷为  0.00 0.00 0.00    [负荷正常]
Entering character mode
Escape character is '^]'


★※※※
```

Enter user name:telenet_user,password:telenet, as diagram 4-19 show:

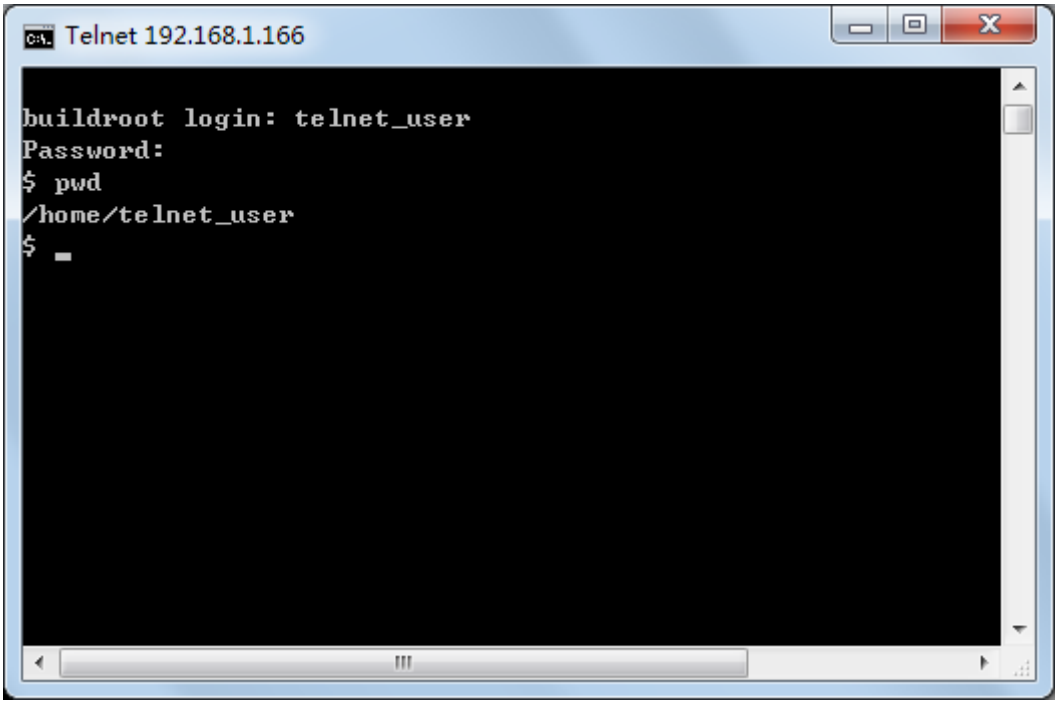


图 4-19

After login on board by telnet, telenet_user can make operations and switch to super user by follow command:

```
$ su -
```

The current directory will change to /root after this command.

Noted: user cann't login by root authority directly, the DVD provide telnent_user in Linux system for login by telenet.

4.7.8 Mount NFS file system

Noted: here we build NFS server on Ubuntu as example.

⑴ Build NFS server on Ubuntu

① install nfs-kernel-server

```
# sudo apt-get install nfs-kernel-server
```

② Add follow content at file end of /etc/exports (The NFS directory is /home/jbo/nfsdir,this directory will be mount on clients)

```
/home/jbo/nfsdir *(rw,sync,no_root_squash)
```

③ Restart portmap and nfs-kernel-server

```
# service portmap restart
# service nfs-kernel-server restart
```

⑵ mount NFS file system on board

① configure IP

```
# ifconfig eth0 192.168.1.166
```

② Make directory

```
# mkdir nfs
```

③ mount NFS file system, The NFS server IP is 192.168.1.223

```
# mount -t nfs -o nolock 192.168.1.223:/home/jbo/nfsdir nfs
```

User can operate nfs's directory, and the change/modification will refect on NFS server /home/jbo/nfsdir.

4.7.9 RTC Use

⑴ Install button battery to board.

⑵ System will set initial value at first start time, so it needs to set the time after system startup.

Set system time:

```
# date -s 2013.06.07-15:51:00 ;
hwclock –w Jul  8 10:36:00 BST 2013
```

Query system time and hardware RTC time:

```
# date
Fri Jun  7 15:52:22 UTC 2013
# hwclock -r
Fri Jun  7 15:53:22 2013  0.000000 seconds
```

4.8 Linux Driver Development Examples

This chapter describes a simple character device driver development, achieving the function to control LED.

4.8.1 Hardware Schematic

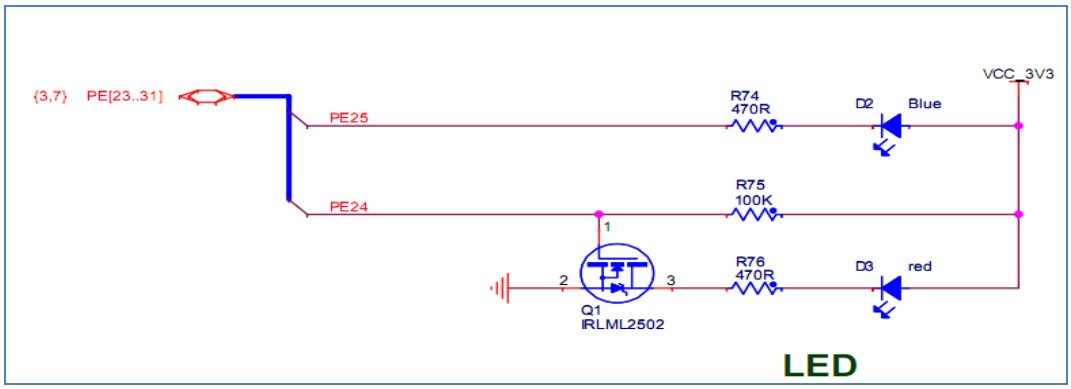


Figure 4-20

Use PD21 interface control D2 by IRLML2502. When it is high, LED turns on. Port PB18 control D1directly. When it is low, LED turns on. Refer to figure 4-20.

4.8.2 Driver Source Code

(1) Create driver file in new kernel

Create driver files in driver/char/:

```
# cd linux-3.6.9
# vi drivers/char/ledtest.c
```

(2) Driver source code ledtest.c is as follows:

```
#include <linux/module.h>
#include <linux/interrupt.h>
#include <linux/gpio.h>
#include <linux/input.h>
#include <linux/miscdevice.h>
#include <asm/uaccess.h>

#define DEVICE_NAME     "gpiokeys"

static struct input_dev *dev_input;
```

```
#define PIN_MYD_KEY3 AT91_PIN_PE27    //PB_USER1
#define PIN_MYD_LED2 AT91_PIN_PE25    //blue
#define PIN_MYD_LED3 AT91_PIN_PE24    //red
#define PIN_MYD_KEY1 AT91_PIN_PC26
#define PIN_MYD_KEY2 AT91_PIN_PC27

static ssize_t dev_read(struct file *filp, char *buffer, size_t count, loff_t *ppos)
{
        int ret;
        char value;
        if (count < 1) return -EINVAL;
        value = 0x30;
        value |= gpio_get_value(PIN_MYD_KEY3);
        ret = copy_to_user(buffer, &value, count);
        return (ret) ? -EFAULT : 1;
}

static ssize_t dev_write(struct file *filp, const char *buffer, size_t count, loff_t *off)
{
        u8 value;
        if (count != 1) return -EINVAL;
        if (copy_from_user(&value, buffer, count))
        {
                pr_err("%s : failed to copy_from_user\n", __func__);
                return -EFAULT;
        }
        gpio_set_value(PIN_MYD_LED2, ((value>>0) & 1));
        gpio_set_value(PIN_MYD_LED3, ((value>>1) & 1));
        return 0;
}
```

```
static int dev_open(struct inode *inode, struct file *filp)
{
	return 0;
}

typedef struct {
	int	gpio;
	int	key0, key1;
	int	count;
	struct timer_list timer;
} gpio_button_data;

// 15: TAB, 102: HOME; 103: UP; 107: END; 108: DOWN; 139: MENU; 158: BACK
static gpio_button_data buttons[]={
	{PIN_MYD_KEY3, 15, 35},		// BACK, HOME
	{PIN_MYD_KEY2, 15, 35},		// BACK, HOME
	{PIN_MYD_KEY1, 15, 35},		// BACK, HOME
};

static void gpiokeys_check(unsigned long a)
{
	u8 value;
	int i = (int)a;
	if (i<0 || i>=sizeof(buttons)/sizeof(buttons[0])) return;
	value = gpio_get_value(buttons[i].gpio);
	printk("gpiokeys %d count=%d value=%d\n", buttons[i].gpio, buttons[i].count, value);
	if (value) // button up
	{
		if (buttons[i].count >= 5 && buttons[i].count < 20)  // quick press
```

```
            {
                input_event(dev_input, EV_KEY, buttons[i].key0, 1);
                input_sync(dev_input);
                input_event(dev_input, EV_KEY, buttons[i].key0, 0);
                input_sync(dev_input);
            }
            buttons[i].count = 0;
            //del_timer(&buttons[i].timer);
        }
        else // button down
        {
            buttons[i].count = buttons[i].count + 1;
            if (buttons[i].count >= 20)    // long press
            {
                buttons[i].count = 0;
                input_event(dev_input, EV_KEY, buttons[i].key1, 1);
                input_sync(dev_input);
                input_event(dev_input, EV_KEY, buttons[i].key1, 0);
                input_sync(dev_input);
            }
            else
            {
                //mod_timer(&buttons[i].timer, jiffies + msecs_to_jiffies(100)); //0.1S
            }
        }
}

/*!
 * gpiokeys interrupt handler.
 */
```

```
static irqreturn_t gpiokeys_int(int irq, void *dev_id)
{
	int n = (int)dev_id;
	gpiokeys_check(n);
	return IRQ_HANDLED;
}

static struct file_operations dev_fops = {
	owner:	THIS_MODULE,
	open:	dev_open,
	read:	dev_read,
	write:	dev_write
};

static struct miscdevice misc = {
	.minor = MISC_DYNAMIC_MINOR,
	.name = DEVICE_NAME,
	.fops = &dev_fops,
};

static int gpio_request_check(unsigned gpio, const char *label)
{
	int ret;
	ret = gpio_request(gpio, DEVICE_NAME);
	if (ret)
		pr_err("gpio %d request failed: %s\n", gpio, label);
	return ret;
}

/*!
```

```
 * This function is called when the module is loaded.
 */
static int __init gpiokeys_init(void)
{
	int i, ret;
	pr_info("gpiokeys init\n");

	dev_input = input_allocate_device();
	if (!dev_input)
	{
		pr_err("%s: not enough memory for input device\n", DEVICE_NAME);
		return -ENOMEM;
	}
	dev_input->name = DEVICE_NAME;
	dev_input->id.bustype = BUS_HOST;
	// Enable auto repeat feature of Linux input subsyste
	__set_bit(EV_REP, dev_input->evbit);

	//dev_input->evbit[0] = BIT_MASK(EV_KEY) | BIT_MASK(EV_REP);

	for (i=0; i<sizeof(buttons)/sizeof(buttons[0]); i++)
	{
		int gpio= buttons[i].gpio;
		int irq = gpio_to_irq(gpio);
		gpio_request_check(gpio, "KEY");
		//gpio_direction_input(gpio);
		at91_set_gpio_input(gpio, 1);    // input with pullup
		gpio_set_debounce(gpio, 10*1000);
		ret = request_irq(irq, gpiokeys_int, IRQF_TRIGGER_FALLING, DEVICE_NAME,
(void *)i);
```

```
            if (ret)
            {
                pr_info("request gpiokeys[%d] irq:%d failed\n", i, irq);
                input_free_device(dev_input);
                return ret;
            }
            input_set_capability(dev_input, EV_KEY, buttons[i].key0);
            input_set_capability(dev_input, EV_KEY, buttons[i].key1);
            //setup_timer(&buttons[i].timer, gpiokeys_check, (unsigned long)i);
        }

        gpio_request_check(PIN_MYD_LED2, "LED2");
        gpio_request_check(PIN_MYD_LED3, "LED3");
        gpio_direction_output(PIN_MYD_LED2, 1);
        gpio_direction_output(PIN_MYD_LED3, 1);

        ret = input_register_device(dev_input);
        if (ret)
        {
            pr_info("error register input device: %d\n", ret);
            input_free_device(dev_input);
            return ret;
        }

        return misc_register(&misc);
}

/*!
 * This function implements the exit function of the device.
 */
```

```
static void __exit gpiokeys_exit(void)
{
	int i;
	gpio_free(PIN_MYD_LED2);
	gpio_free(PIN_MYD_LED3);

	input_unregister_device(dev_input);
	for (i=0; i<sizeof(buttons)/sizeof(buttons[0]); i++)
	{
		int gpio= buttons[i].gpio;
		int irq = gpio_to_irq(gpio);
		free_irq(irq, (void *)i);
		//del_timer_sync(&buttons[i].timer);
		gpio_free(gpio);
	}
	misc_deregister(&misc);
}
module_init(gpiokeys_init);
module_exit(gpiokeys_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Sherwin lee.");
MODULE_DESCRIPTION("MYD-SAMA5D3 gpiokeys device driver");
```

4.8.3 Compile the Driver

(1) After Install and Compile Linux kernel Source Code, Modify Makefile's KDIR point to the linux source path(Here is: /opt/linux/ Linux-at91/linux-at91):

```
	# cd /opt/linux/gpio-source
	# tar xvjf gpioleds.tar.bz2
```

```
# cd gpioleds
# sudo vi Makefile
KDIR:= /opt/linux/ Linux-at91/linux-at91
```

Then using the command: wq keep quit

(2) Modify corresponding variable path:

```
# export ARCH=arm
# export
CROSS_COMPILE=/opt/gcc-linaro-arm-linux-gnueabihf-4.7-2013.04-20130415_linu
x/bin/arm-linux-gnueabihf-
```

(3) Compile and ls:

```
# make
# ls
```

After compile, gpioleds.ko in /home/myir is Linux Driver file that we need.

4.9 Application Development Instance

This chapter describes the upper layer of the Linux system application development, and a simple instance tells the application development process and driver invocation. Instance to achieve the function: when run application, board can control two bright LED and specific LED lights on or off is controlled by passed parameters.

4.9.1 Download Driver into Board

Copy gpiokeys.ko to SD card or U disk, which is loaded in the corresponding directory, specific actions are as follows:

```
# mount -t vfat /dev/sda1 /mnt
# cd /mnt
# ls
gpioleds.ko
# insmod gpioleds
```

At this point, LED driver has loaded into kernel successfully.

4.9.2 Driving test

(1) Turn on blue LED

```
# echo 1 >/sys/class/leds/blue/brightness
```

(2) Turn off blue LED

```
# echo 0 >/sys/class/leds/blue/brightness
```

(3) Turn on red LED

```
# echo 1 >/sys/class/leds/red/brightness
```

(4) Turn off red LED

```
# echo 0 >/sys/class/leds/red/brightness
```

(5) System runs with red LED indicator light

```
# echo heartbeat >/sys/class/leds/red/trigger
```

4.10 Qt use Guide

This section describe the methods and steps to using Qt GUI application development on the MYD - SAMA5D3X, including two parts, the first tells the CD provided in the use of the Qt cross compiler tool chain, the general Qt application development use the CD provided in the Qt cross tool chain; The second part tells how to compile from Qt - Embedded source code generated Qt cross tool chain and related library file, when the CD provided in the Qt library can't meet the demand of Qt development program only need to make Qt development environment.

Note: on the development board to download 02-Im ages/Linux directory of Images, for example, download specific steps refer to automatically download or 4.3. 3, 4.3. 4 manually down -load. Here for Ubuntu10.04 PC environment.

4.10.1 Using the Qt cross compiler tool chain

For normal Qt application development can be directly use CD available in Qt cross tool chain, in the disc 02-Im ages/Linux/Qt_Arm directory file already contains the tool chain corresponding Qt libraries, so use This tool chain compiled Qt applications can run

on the board directly. Disc available in Qt cross tool chain configuration in detail Shown in the following table:

Configuration items	Value
Build	libs
Debug	no
Qt 3 compatibility	yes
QtDBus module	no
Qt ScriptTools module	yes
Qt XmlPatterns module	no
Phonon module	no
SVG module	yes
WebKit module	yes
STL support	yes
PCH support	yes
MMX/3DNOW/SSE/SSE2	no/no/no/no
iWMMXt support	no
IPv6 support	yes
IPv6 ifname support	yes
getaddrinfo support	yes
getifaddrs support	yes
Accessibility	yes
NIS support	yes
CUPS support	no
Iconv support	no
Glib support	no
GStreamer support	no
Large File support	yes
GIF support	plugin

Configuration items	Value
TIFF support	plugin (qt)
JPEG support	plugin (qt)
PNG support	yes (qt)
MNG support	plugin (qt)
zlib support	yes
Session management	no
Embedded support	arm
Freetype2 support	yes
Graphics (qt)	linuxfb multiscreen linuxfb
Graphics (plugin)	
Decorations (qt)	styled windows default
Decorations (plugin)	
Keyboard driver (qt)	tty usb
Keyboard driver (plugin)	
Mouse driver (qt)	pc linuxtp pc linuxtp tslib
Mouse driver (plugin)	
OpenGL support	no
SQLite support	qt (qt)
OpenSSL support	no

(1) Install Qt cross compile tools to system set up under the/usr/local/directory of Qt - arm work directory, copy Qt cross compile tools to this directory and unzip the:

```
$ cd ~
$ mkdir qt-arm
$ cd qt-arm
$ cp /media/cdrom/05-Linux_Source/Qt_Arm/Qt-4.5.3_Tslib-1.4.tar.gz ./
$ sudo tar xvzf Qt-4.5.3_Tslib-1.4.tar.gz –C /usr/local/
```

Decompression is completed, / usr/local directory will be more than two directories:

qt4.5.3-arm and tslib, including qt4.5.3 - arm directory contains cross-compilation Qt application use of tools, libraries and header files, etc.; Tslib test program directory contains the touch screen, used link libraries, configuration files, etc.

(2) set the system environment variable

① If you haven't added the arm-Linux- gnueabi - PATH to the PATH of cross compiler, this article USES the cross compiler PATH is/usr/local/gcc-linaro-arm-linux-gnueabihf-4.7-2013.04-20130415_linux/bin,execute the following command:

```
$ export
PATH=$PATH:/opt/gcc-linaro-arm-linux-gnueabihf-4.7-2013.04-20130415_linux/bin
```

② Set the environment variable Qt application development

Execute the following command at terminal:

```
$ export PATH=$PATH:/usr/local/qt4.5.3-arm/bin
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/qt4.5.3-arm/lib
```

Or add the above command to the/etc/profile file, so set these environment variables automatically when system login.

(3) Burning Qt library image

Reference 4.3.3 automatically download or 4.3.4 manually download to burning CD 02-Im ages/Linux/Qt_Arm directory of Images, the Images in the directory has been set for Qt runtime environment, and already contains the Qt application runtime Commonly used libraries, supported by the function modules as shown in table 4-2. If the features provided by the mirror can't meet the actual demand,Please refer to the next section 4.10.2 cross-compilation Qt development environment from the source code to configure the required function modules.

(4) Compile Qt and run the application

There are several Qt sample program in 05 - Linux_Source/Qt_Arm/ Qt_Examples/, in here Masterdetail for example, using Qt cross-compilation toolchain compile Qt application methods, and run on the target board.

① Copy the sample program to Qt - arm

```
$   cd ~/qt-arm
$ cp /media/cdrom/05-Linux_Source/Qt_Arm/Qt_Examples/masterdetail.tar.gz ./
```

```
$ tar xvzf masterdetail.tar.gz
$ cd masterdetail
```

② Compile the Qt project :

```
$ qmake -project
$ qmake
$ make
```

If compilation error: QtSql (QtXml) to a file or directory, the need to manually edit after execution of qmake project - masterdetail. Pro file, add in the file: QT + = SQL (XML).

③ After perform the above command, the generated executable file masterdetail can copy to the development board to run on:

```
# ls
masterdetail
# chmod 0777 masterdetail
# ./masterdetail -qws
```

The results as shown in the figure 4-21 below:

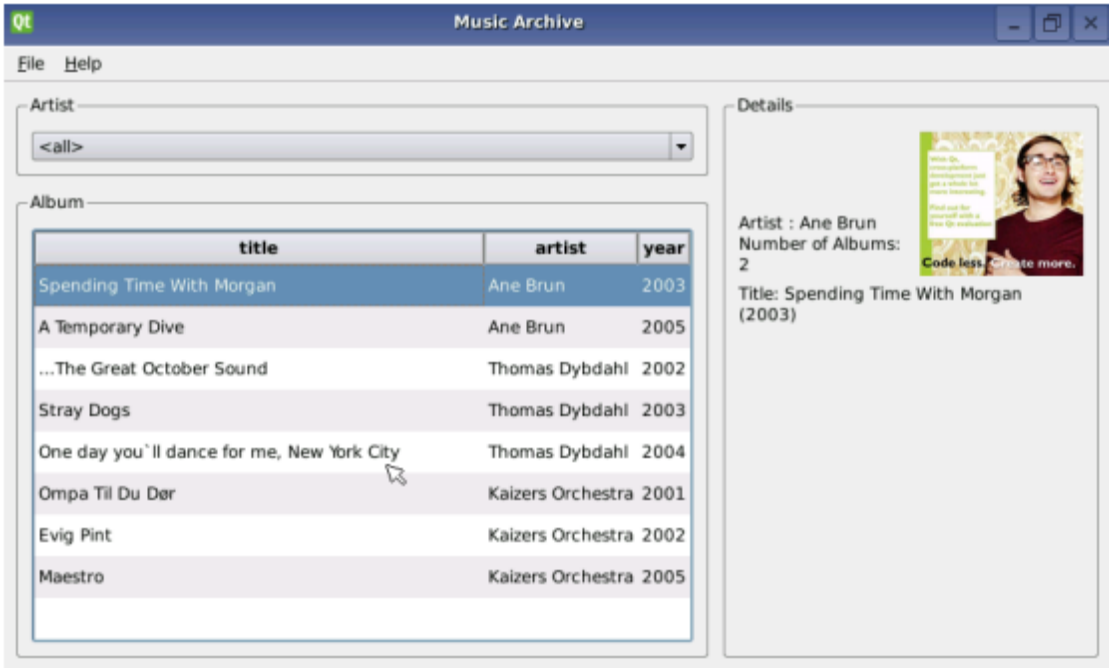


Figure 4-21

4.10. 2 Qt development environment

This section describes how to compile from Qt - Embedded source code generated Qt cross tool chain and related arm - the Qt libraries.When the CD provided in the Qt library can not meet the demand of actual development can through this method when making Qt development and runtime environment.Qt source code and the Tslib source code are in02-Im ages/Linux/Qt_Arm CD catalogue.

⑴ Establish a working directory

```
$ cd ~
$ mkdir qt-arm
$ cd qt-arm
```

⑵ Compile tslib installation

① Unzip:

```
$ cp /media/cdrom/05-Linux_Source/tslib.tar.gz ./
$ tar xvzf tslib.tar.gz
$ cd tslib
```

② Compile installation:

If you haven't added the arm - none - Linux - gnueabi - cross compiler PATH to the PATH, to look over the PATH Added to the PATH, this article USES the cross compiler PATH is/usr/local/GCC-linaro-arm-inux_linux gnueabihf-4.7-2013.04-20130415/bin, perform the following command:

```
$ export PATH=$PATH:/opt/
gcc-linaro-arm-linux-gnueabihf-4.7-2013.04-20130415_linux/bin
```

Install automake, libtool first two tools:

```
$ sudo apt-get install automake libtool
```

Configuration tslib installation path, can set themselves up to/usr/local/tslib installation:

```
$ ./autogen.sh
$ ./configure CC=arm -linux-gnueabi-gcc CXX=arm-linux-gnueabi-g++
--prefix=/usr/local/tslib --host=arm-linux ac_cv_func_malloc_0_nonnull=yes
```

Compile the installation:

```
$ make
```

```
$ sudo make install
```

After installing the tslib to tslib/etc/ts. The conf file the second line "# module_raw input" annotation, into "module_raw input", attention must thus.

(3) Compile install qt - embedded:

① Unzip:

```
$ cp /media/cdrom/05-Linux_Source/Qt_Arm/Qt_Source
/qt-embedded-linux-opensource-rc-4.5.3.tar.gz ./
$ tar xvzf qt-embedded-linux-opensource-src-4.5.3.tar.gz
$ cd qt-embedded-linux-opensource-src-4.5.3
```

② Specify cross-compiler:

Open the mkspecs/QWS/Linux - arm - g + + / qmake. Conf:

```
$ vi mkspecs/qws/linux-arm-g++/qmake.conf
```

After using vi qmake. Conf, enter the following command, arm-Linux-all with a replacement for Arm-none-Linux-gnueabi-and add-LTS, then save, changes are as follows:

```
# modifications to g++.conf
QMAKE_CC = arm-linux-gnueabi-gcc -lts
QMAKE_CXX = arm-linux-gnueabi-g++ -lts
QMAKE_LINK = arm-linux-gnueabi-g++ -lts
QMAKE_LINK_SHLIB = arm-linux-gnueabi-g++ -lts
```

③ Qt configuration:

```
$ ./configure -prefix /usr/local/qt4.5.3-arm -xplatform qws/linux-arm-g++ -release
-opensource -qt-zlib -qt-libtiff -qt-libpng -qt-libmng -qt-libjpeg -make libs -nomake docs
-embedded arm -little-endian -qt-freetype -depths 8,16,24 -qt-gfx-linuxfb
-qt-kbd-usb -qt-mouse-pc -qt-mouse-linuxtp -qt-mouse-tslib -qt-sql-sqlite
-qt3support -I/usr/local/tslib/include -L/usr/local/tslib/lib -confirm-license
```

Perform / configure -- help to see detailed descriptions of the parameters, according to the need to configure the appropriate parameters.

④ Compile the installation:

```
$ make
```

```
$ sudo make install
```

⑤ Set the environment variables:

Execute the following command at terminal:

```
$ export PATH=$PATH:/usr/local/qt4.5.3-arm/bin
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/qt4.5.3-arm/lib
```

Or add the above command to the/etc/profile file, so set these environment variables automatically when system login.

(4) Transplante Qt to development board

① Copy library to the board

Qt after installation is complete, in order to solve the problems of the symbolic link, can use package first, then uncompress directly to the development board:

```
$ cd    /usr/local/qt4.5.3-arm
$ tar    -zcf    lib.tar.gz    lib
```

Copy lib.tar.gzto the development board, and then unpack:

```
# mkdir -p /usr/local/qt4.5.3-arm
# tar xzvf lib.tar.gz -C /usr/local/qt4.5.3-arm
```

② Set up the development environment variable

Environment variable is set in the development board has been written in the/etc/setqtenv file, execute the following command on the development board Can complete the environment variables set

```
# source    /etc/setqtenv
```

Or manually input Settings:

```
# export LD_LIBRARY_PATH=/lib:$LD_LIBRARY_PATH
# export QTDIR=/usr/local/qt4.5.3-arm
# export T_ROOT=/usr/local/tslib
# export PATH=$T_ROOT/bin:$PATH
# export LD_LIBRARY_PATH=$T_ROOT/lib:$QTDIR/lib:$LD_LIBRARY_PATH
# export TSLIB_CONSOLEDEVICE=none
# export TSLIB_FBDEVICE=/dev/fb0
# export TSLIB_TSDEVICE=/dev/input/event0
```

```
# export TSLIB_PLUGINDIR=$T_ROOT/lib/ts
# export TSLIB_CONFFILE=$T_ROOT/etc/ts.conf
# export TSLIB_CALIBFILE=/etc/pointercal
# export QWS_KEYBOARD=USB:/dev/input/event1
# export QWS_MOUSE_PROTO=Tslib:/dev/event0
# export QT_QWS_FONTDIR=$QTDIR/lib/fonts
```

Chapter 5 Android System Guide

5.1 Overview

Android is a Linux system based open source operating system, mainly used in portable devices. Android operating system originally developed by Andy Rubin development, initially mainly support mobile phone. In 2005 Android is acquainted by Google, formatting the open mobile phone alliance to improvement it, gradually extended to the tablet computer and other area. Since its first release Welcomed by the majority of consumers, Android's market shares around the world more than Symbian system for the first time in the first quarter of 2011, ranking first in the world. The data shows that in February 2012, Android accounted for 52.5% of the share of the global smartphone operating system market.

Android system is running based on Linux system, mainly made by Linux Kernel, system libraries, Dalvik virtual machine, application framework, and applications written mainly by JAVA. Its framework is as shown in figure 5-1:

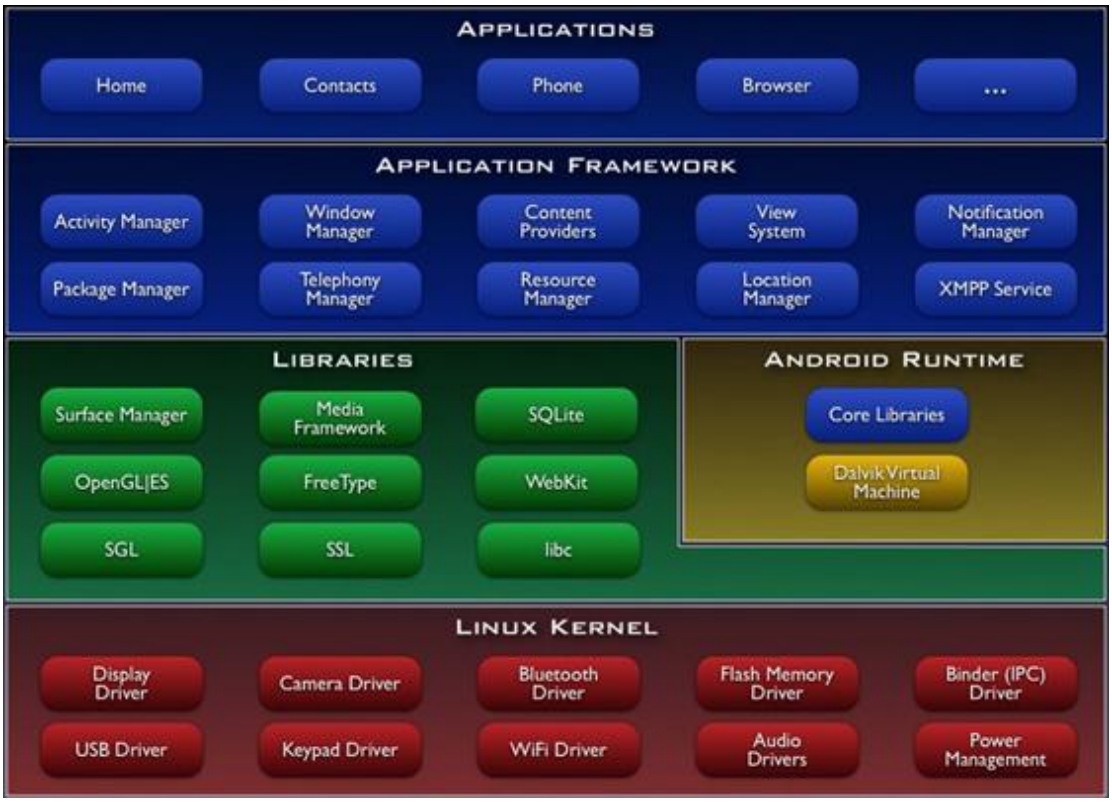


Figure 5-1

This chapter describes how to build and run Android 4.0.4 system in MYD-SAMA5D3X platform, include the following main content:

(1) Build Android system

(2) Compile Android

(3) Android System use

5.2 Software Resources

Software resources are shown in table 5-1

Category	Name	Note
Boot program	AT91Bootstrap	Use to guide Uboot
	Uboot	1.Support Nand-Flash Erase, read and write 2. Support network to download image 3. Support settings, save the environment variable 4. Support display, contrast, modify memory content 5. Support the bootm, bootargs settings
Linux Kernel	Linux 3.6.9	Develop Linux kernel for MYD-SAMA5D3X hardware
Device Drivers	Network port driver	ETH0
	Serial port driver	USART1,DBGU
	USB	USB_HOST*2,USB_OTG
	SD card driver	Micro-SD,SDCard
	LCD+touch	LCD driver
	GPIO driver	Have been tested to provide the source
System Files	Android System Files	Have been tested to provide binary image file

Table 5-1

5.3 Build Android System

This chapter describes how to use image to build Android system.

5.3.1 Install Download Tool

Install Atmel ISP download software SAM-BA (2.12 or later, Note: If install SAM-BA 2.11 and earlier versions, it needs to first uninstall the all (**SAM-BA and USB driver**). If need two or more SAM-BA version coexistence, different SAM-BA versions use different USB interface.

5.3.2 Connect Board and SAM-BA

⑴ Connect Board to PC (Order cannot be reversed) ,Specific steps are as follows:

① Development board switch to 5 v (power off-position)

② Through the micro-USB line connected to the PC and development board

③ Disconnect the CPU module jumper JP1, JP2 backboard disconnect, hold CS_BOOT button At the same time development board switches to USB_5v. IF first time the PC opportunities prompted to install driver development board, Selected at this time SAM - BA installation directory under the relevant position can be installed as shown in figure 5-2:

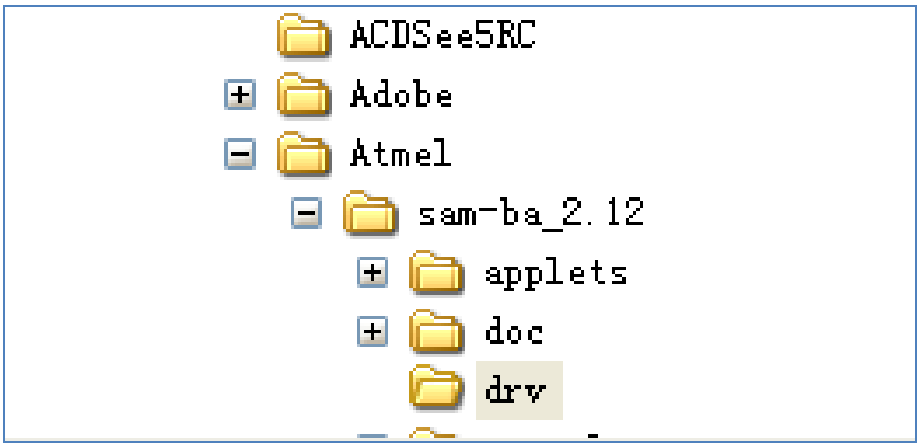


Figure 5-2

⑵ If there is figure 4-3 in "my computer->properties->Management-> device manager->

port", which shows board driver has been installed (According to the actual situation, here for COM4).

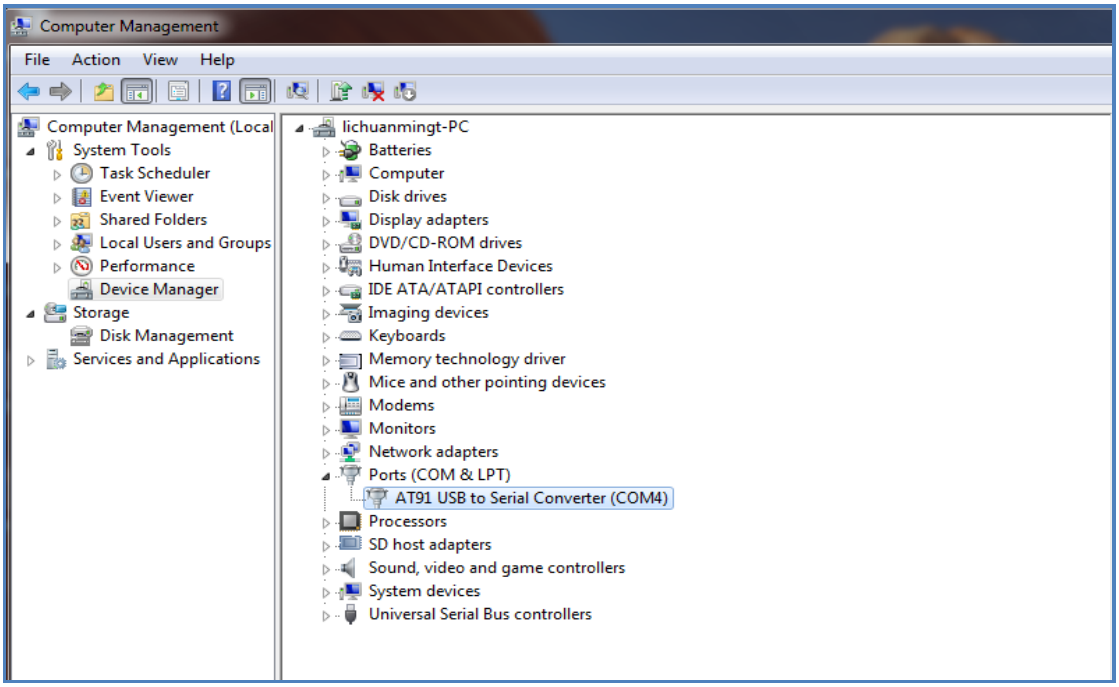


Figure 5-3

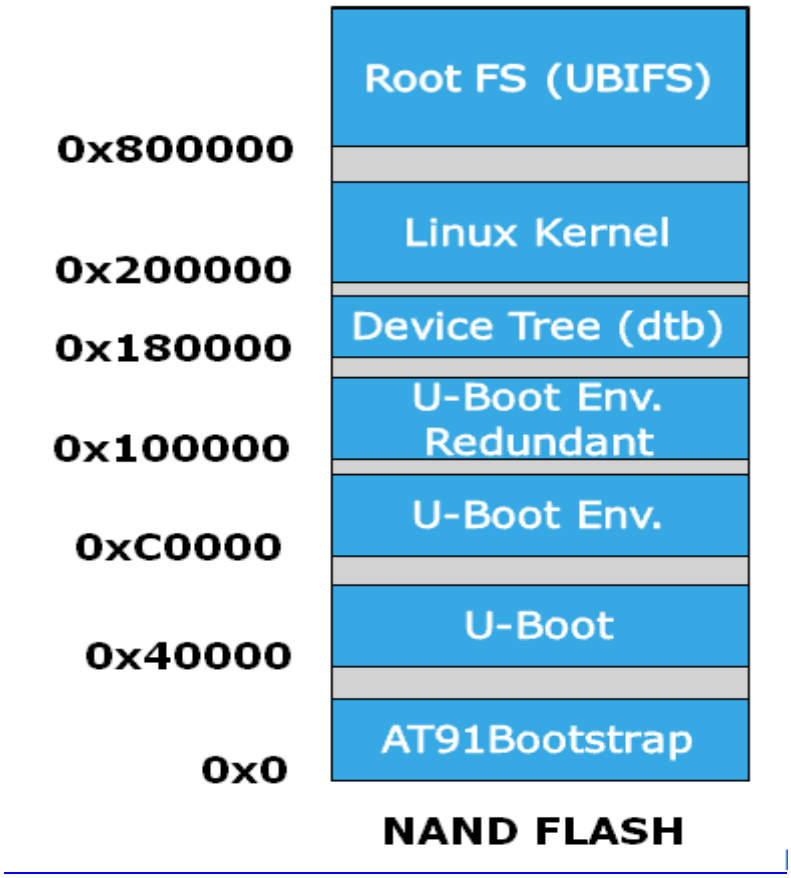
5.3.3 Automatic Download

Note: please pull out SD card before download, otherwise an error may occur

After complete chapter 5.3.1 and 5.3.2, open CD-ROM directory: \Android \demo-image, double-clickflash_all.bat. Then SAM-BA will download Android automatically to board. Entire download process takes about three minutes. When pop logfile.log file automatically, reset board, there will be Android start information.

5.3.4 Manual Download

All image files used in this chapter can be found in the directory: MYIR_SAM A5D3X/02 -Images\Android-image The Nand-Flash content of Android system is divided as shown in figure 5-4:



Fiugre 5-4

Download Linux by SAM-BA manually

(1) Complete chapter 5.3.1 and 5.3.2, double-click samba v2.11, there appears interface. Refer to figure 5-5:

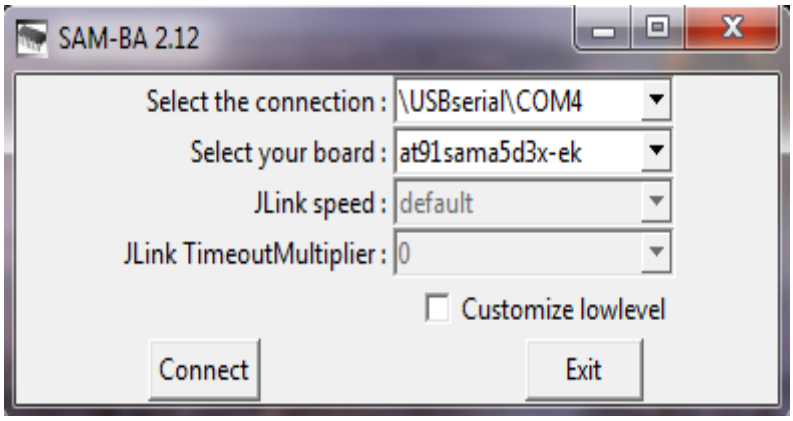


Figure 5-5

Click “Connect” to enter SAM-BA interface. Refer to figure 5-6:

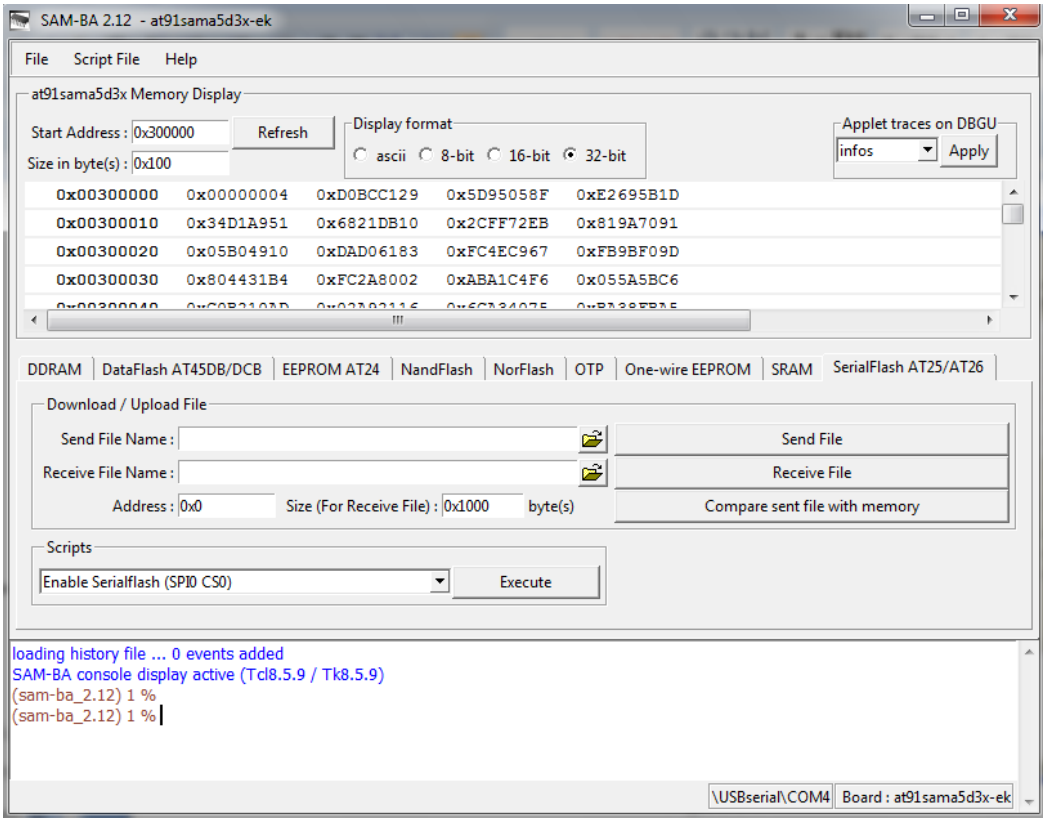


Figure 5-6

(2) Select Nand-Flash tab, Enable Nand-Flash in Scripts tab and then click “Execute”. Refer to figure 5-7:

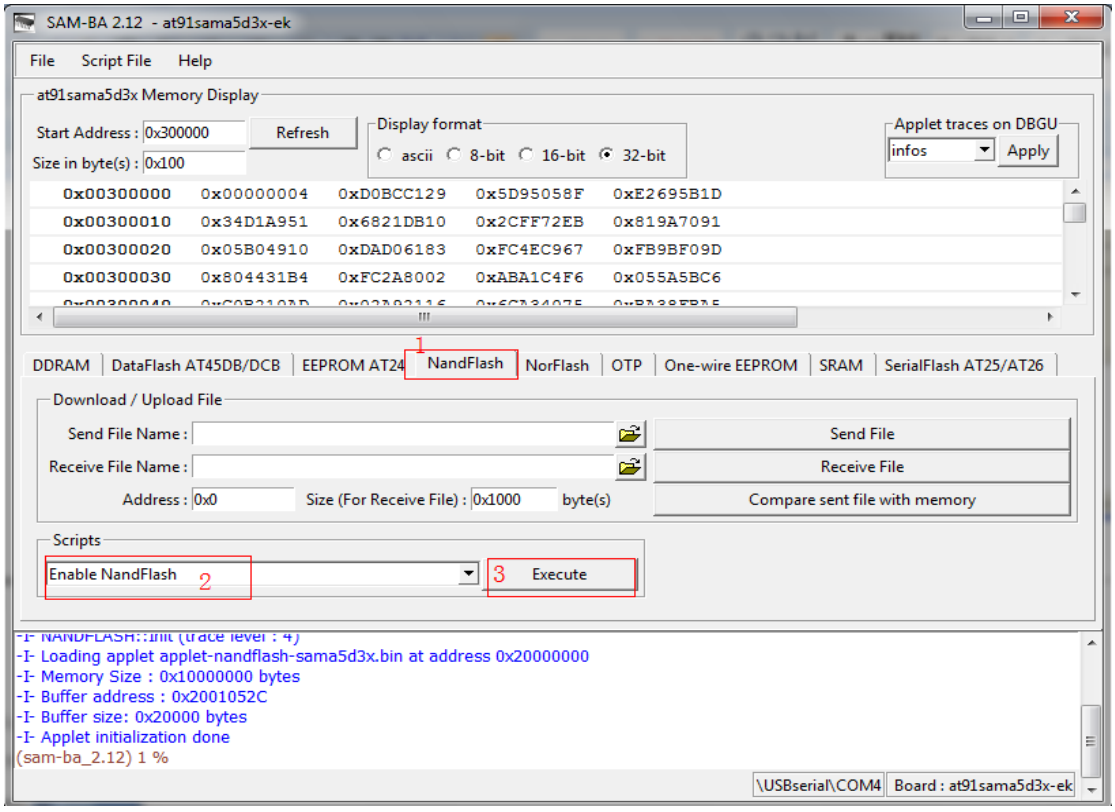


Figure 5-7

(3) Select Enable OS PMECC parameters in Scripts tab, then click Execute, using the default option, click “OK” directly (Note: there cannot check Trimffs). Refer to figure 5-8:

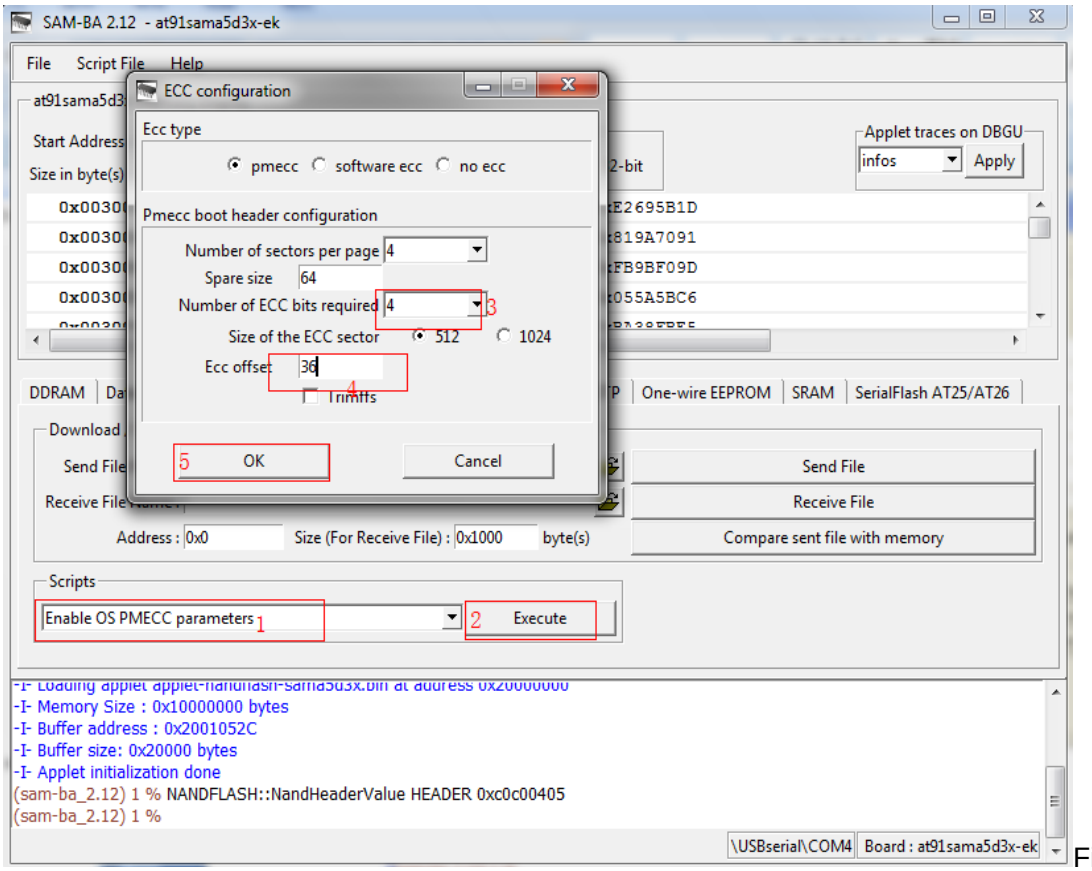


Figure 5-8

(4) Select Erase All in Scripts tab and then click Execute. Refer to figure 5-9:

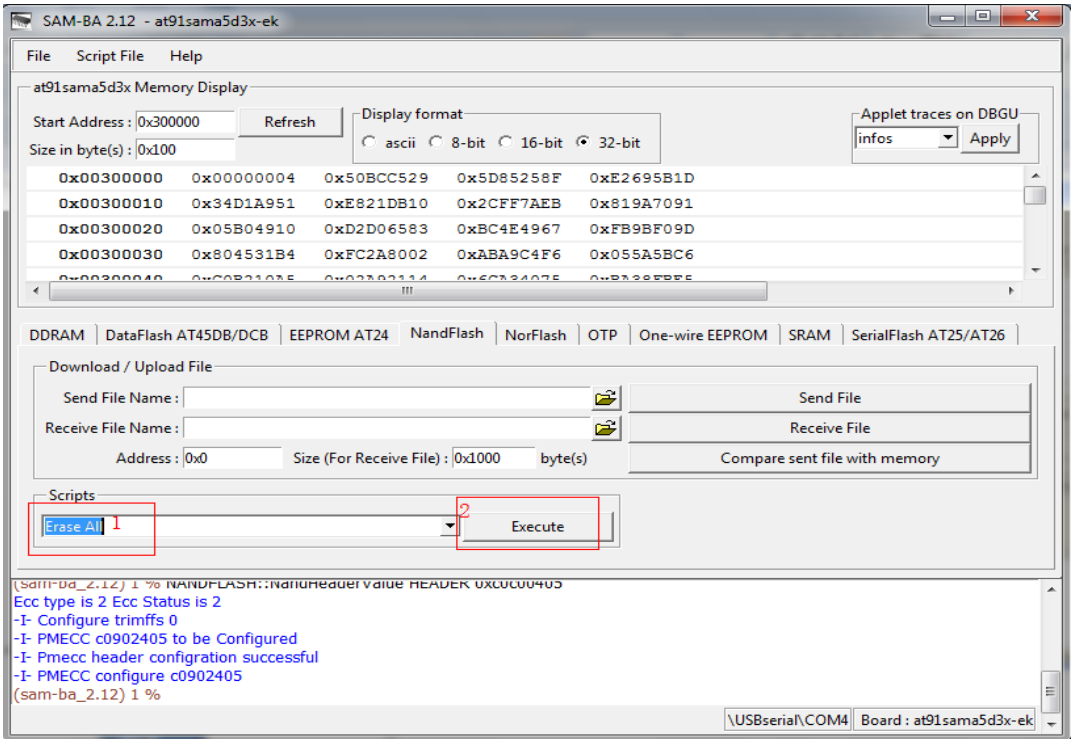


Figure 5-9

(5) Download [boot.bin]. Refer to figure 5-10, 5-11:

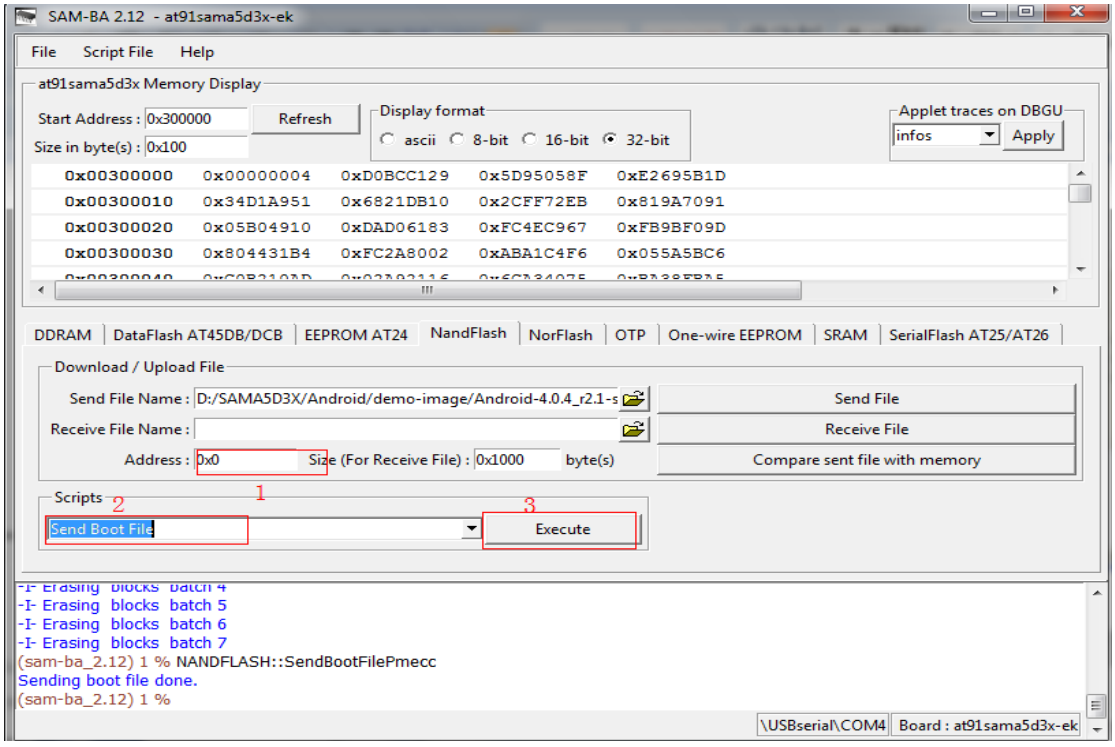


Figure 5-10

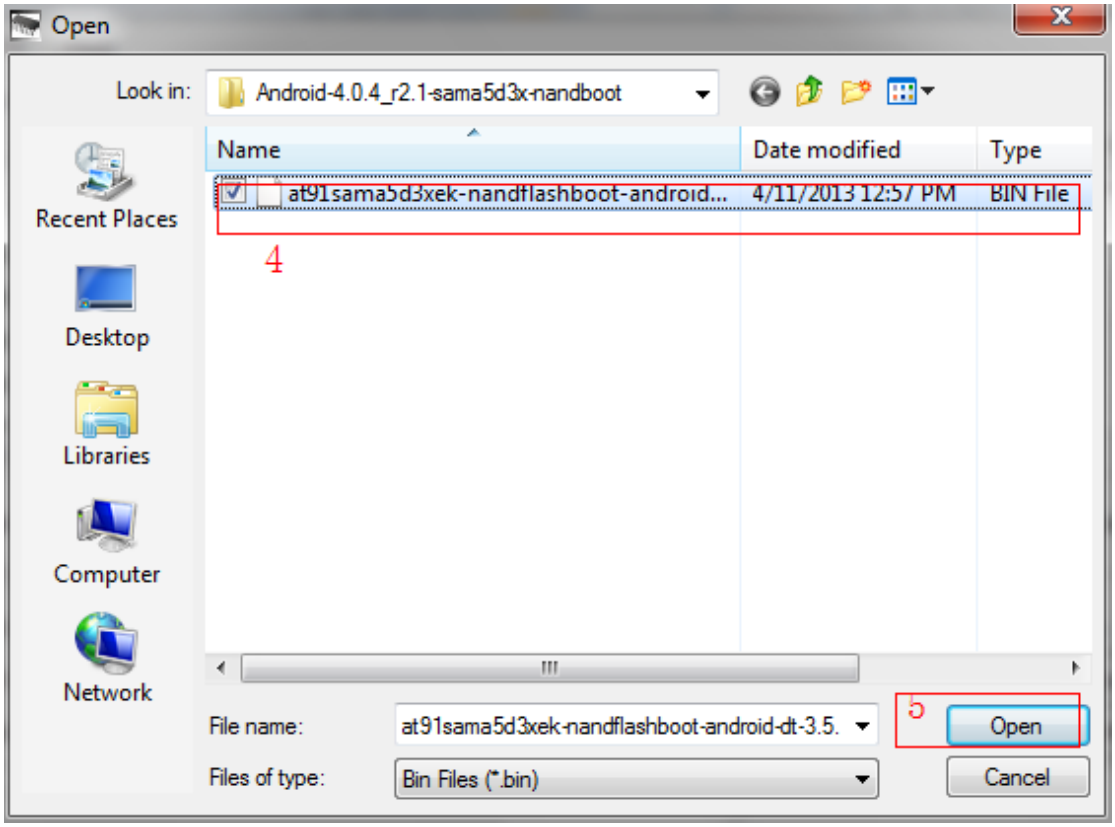


Figure 5-11

⑹ Download [u-boot.bin] to 0x40000. Refer to figure 5-12:

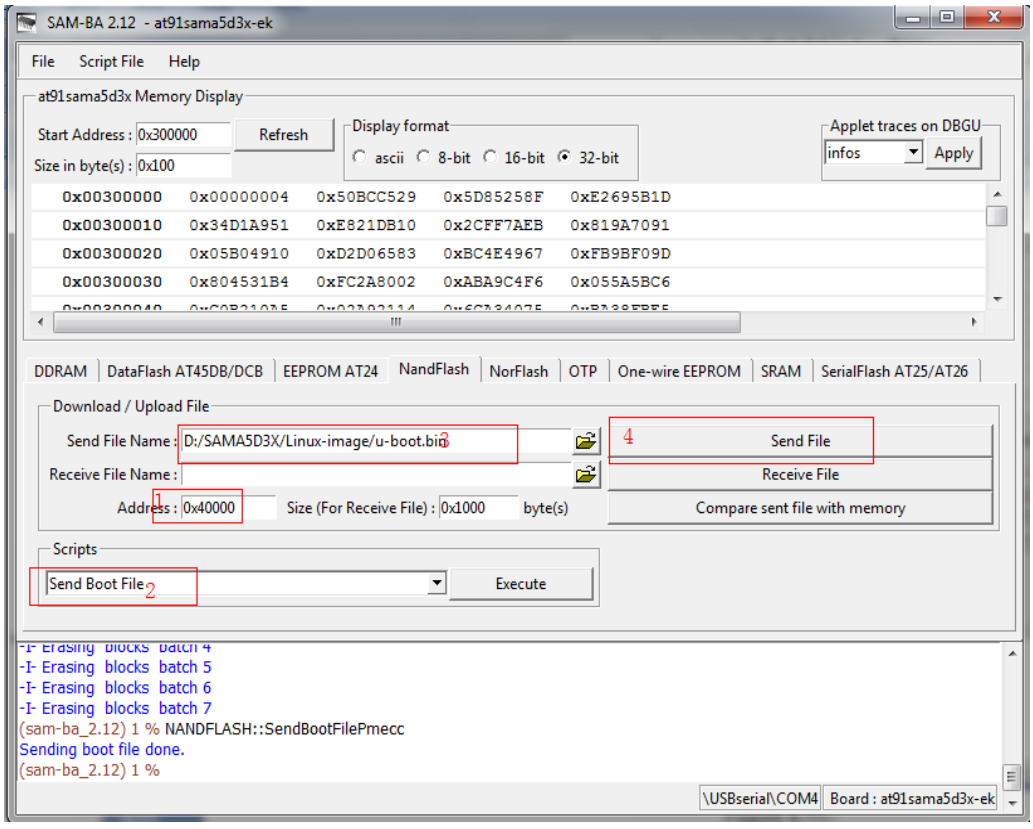


Figure 5-12

(7) Download [ubootEnvtFileNand-Flash.bin] to 0x000C 0000. Refer to figure 5-13:

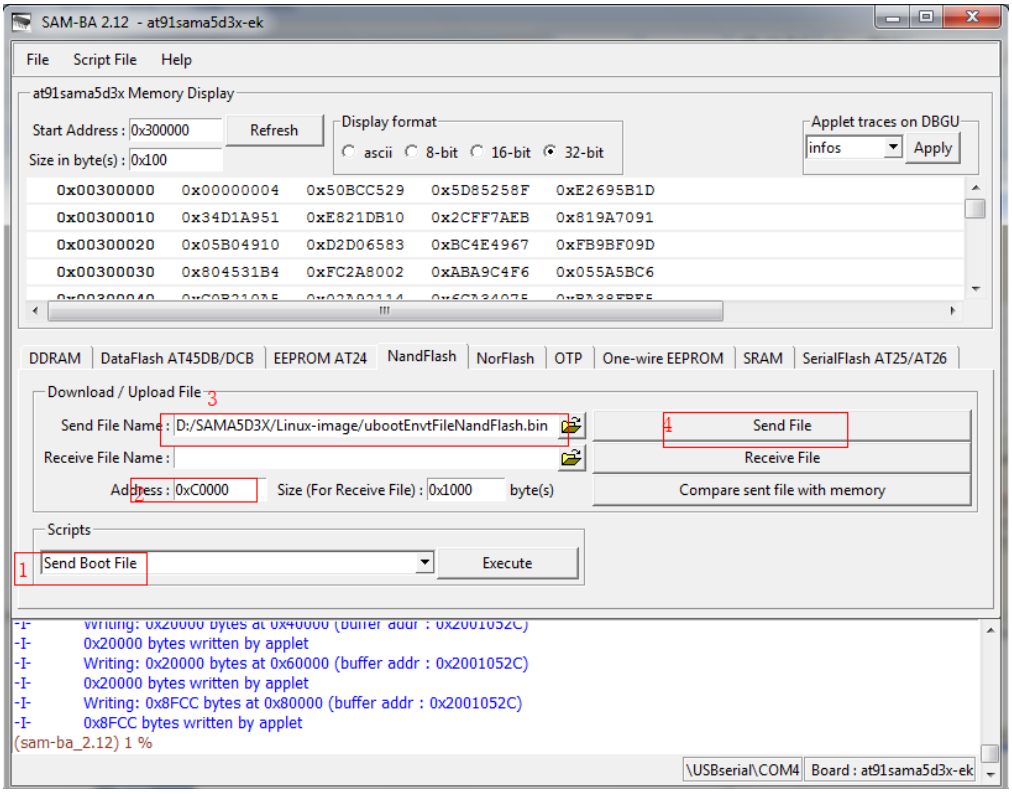


Figure 5-13

(8) Download [sama5d3xek.dtb] to 0x0018 0000 Department. Refer to figure 5-14:

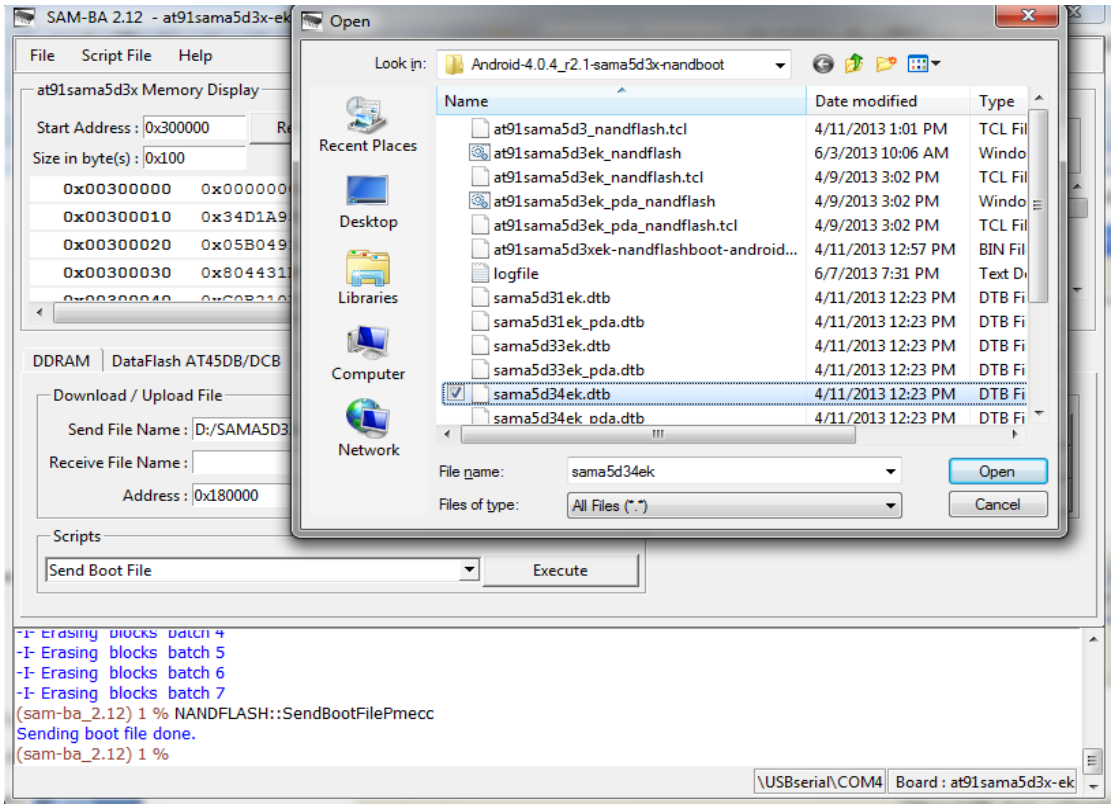


Figure 5-14

⑼ Download Linux kernel [uImage] to 0x0020 0000. Refer to figure 5-15:

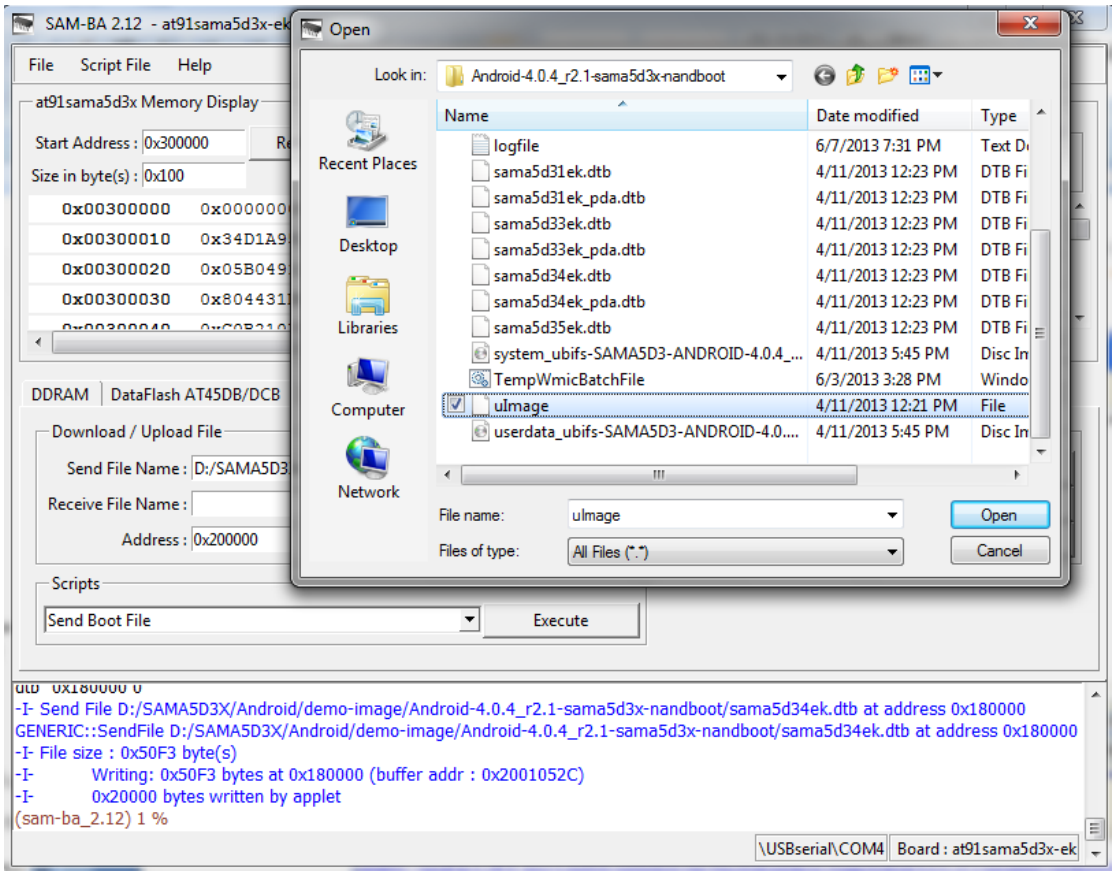


Figure 5-15

⑽ Download [rootfs.ubi] to 0x0050 0000. Refer to figure 5-16(Note: be sure to check Trimffs).

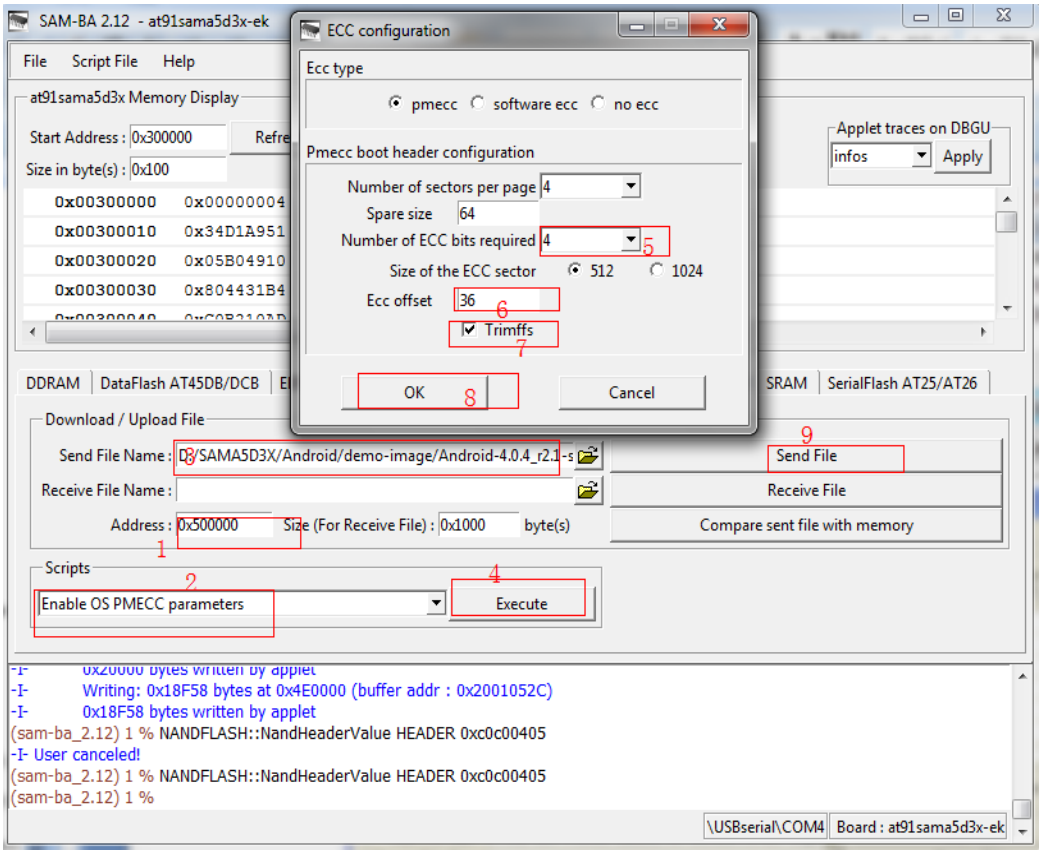


Figure 5-16

At this point, Android system image file download is completed, and press K1(Reset) key can restart Android system.

5.4 Compile Android System Files

This chapter will describe the compiled methods and steps of Android system files.

5.4.1 Android System Principle

(1) File description:

File	Description
boot.bin	Boot program Compiled by AT91 Bootstrap source used to start u-boot
u-boot.bin	The secondary boot for boot kernel
u-boot Env	Configure the environment variables,

	provided u-boot operation parameters
sama5d3x.dtb	Describe hardware device tree, u-boot DTB in the start-up will Flash when the address of the file as the entrance to the parameters passed to the kernel
uImage	Linux kernel file compiled by Linux kernel source code
rootfs.ubi	Android file system
at91SAMA5D3X.tcl	Writing log file and view it by notebook
at91SAMA5D3Xek_demo_android_Nand-Flash.bat	Automatic programming tools (MS - DOS batch file, the manual programming process does not require this file)

Table 5-2

⑵ The principle of the system

Power on, when system starts form Nand-Flash, the start steps is as following:

① Fixed boot code in atsama5d3x internal rom and copy a boot program at91SAMA5D3Xek-Nand-Flashboot-3.1.bin in Nand-Flash to SRAM to run. Boot loader initializes hardware basically, such as setting CPU frequency, config running uration PIO, and then copy the secondary boot program uboot.bin to DDRAM and begin to implement.

② Secondary bootloader uboot is mainly responsible for boot Linux, including set Linux operating environment, Load Linux image file uImage, pass startup parameters to Linux, last boot Linux to start .

③ When boot Linux kernel, Android file system will be mounted automatically. At this point, Android system is booted.

5.4.2 Compile System Files

We know that Android system is running Linux-based system, so if build Android

system, set up a Linux- based platform firstly.

⑴ Decompression cross compiler tool to /usr/local.compile[AT91Bootstrap] to [at91SAMA5D3Xek-Nand-Flashboot-3.1.bin] ,compile u-boot to [u-boot.bin] refer to 【4.5.1】,【4.5.2】,【4.5.3】,【4.5.4】.

⑵ Compile Linux kernell used in Android system

Unzip Linux kernel to working directory:

```
# cp  /media/cdrom/06-Android_Source/Linux-at91/linux-at91.tar.bz2  /opt/android
# cd  /opt/android
# tar  xvjf  linux-at91.tar.bz2
# cd  linux-at91
```

According to actual situation select appropriate configuration files renamed as ".config ":

```
# cp  myd_sama5d3_android_defconfig  .config
```

Compile Linux kernel:

```
# export ARCH=arm
# export CROSS_COMPILE=/opt/
gcc-linaro-arm-linux-gnueabihf-4.7-2013.04-20130415_linux/arm-linux-gnueabi-
# make menuconfig(pops up formulate box directly and save out)
# make
# make uImage
# make dtbs
```

Note: make uImage command requires compile environment installed uboot-mkimage tool; otherwise, use the following command to install tool:

apt-get install uboot-mkimage

After compile kernel, uImage in directory arch/arm/boot/ is Linux kernel programming file. sama5d31ek.dtb, sama5d33ek.dtb, sama5d34ek.dtb, sama5d35ek.dtb is Android device tree image file what we need.

5.5 Android System Use

5.5.1 USB Keyboard Test

Insert USB keyboard to J24, press the Num Lock key, when the lights in the upper-right corner turn green, it shows usb keyboard can be used as far as on PC to input

5.5.2 Browse Picture Test

(1) Select "Gallery" icon, Refer to figure 5-15:

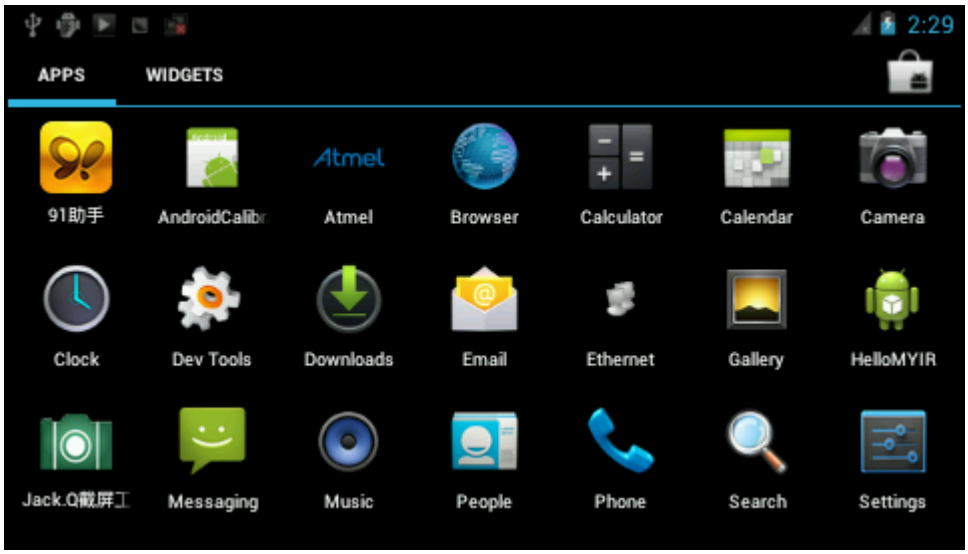


Figure 5-15

(2) Click “Gallery” and select a picture folder, Refer to figure 5-16:

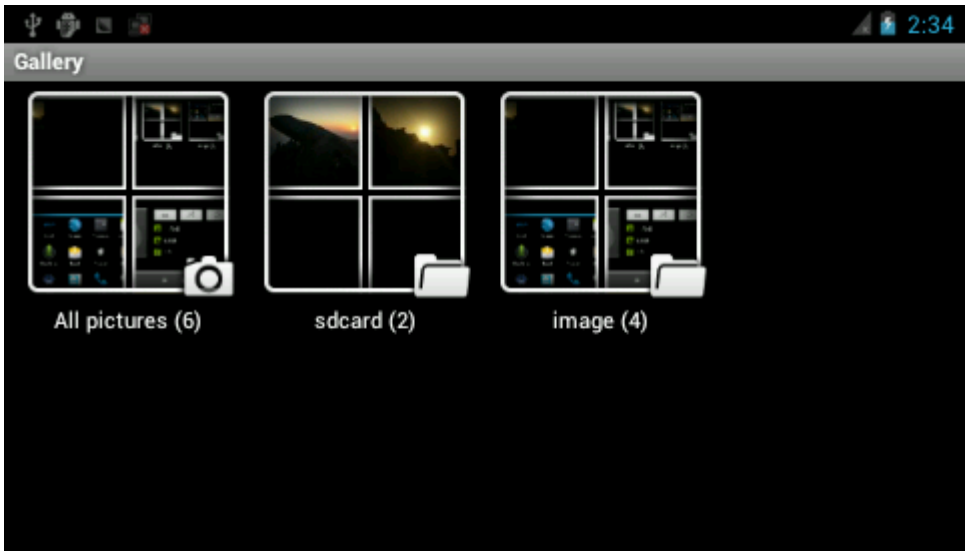


Figure 5-16

(3) View picture, the results are shown in figure 5-17:

Figure 5-17

5.5.3 Play Audio Test

(1) Enter interface and Select "Music" icon, Refer to figure 5-18:

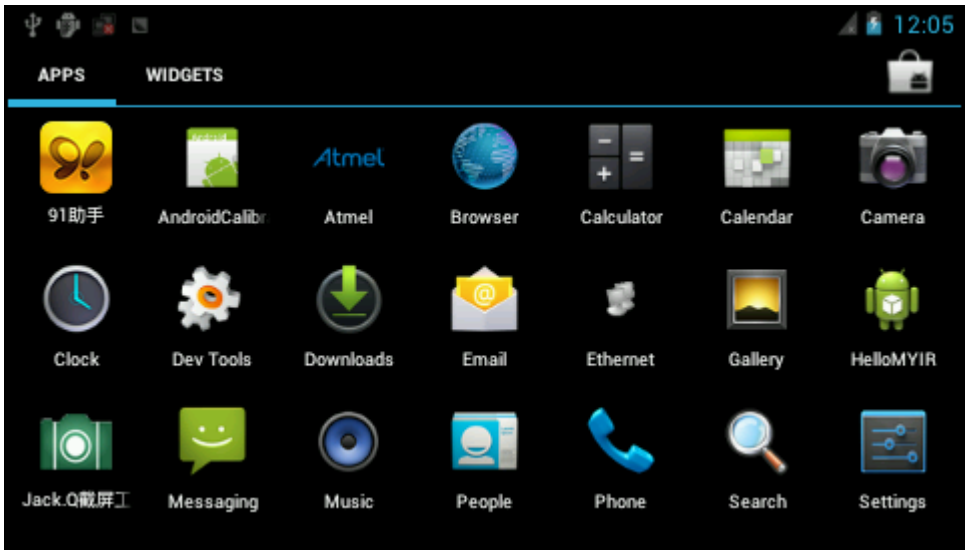


Figure 5-18

(2) Select "Songs" option. Refer to figure 5-19:

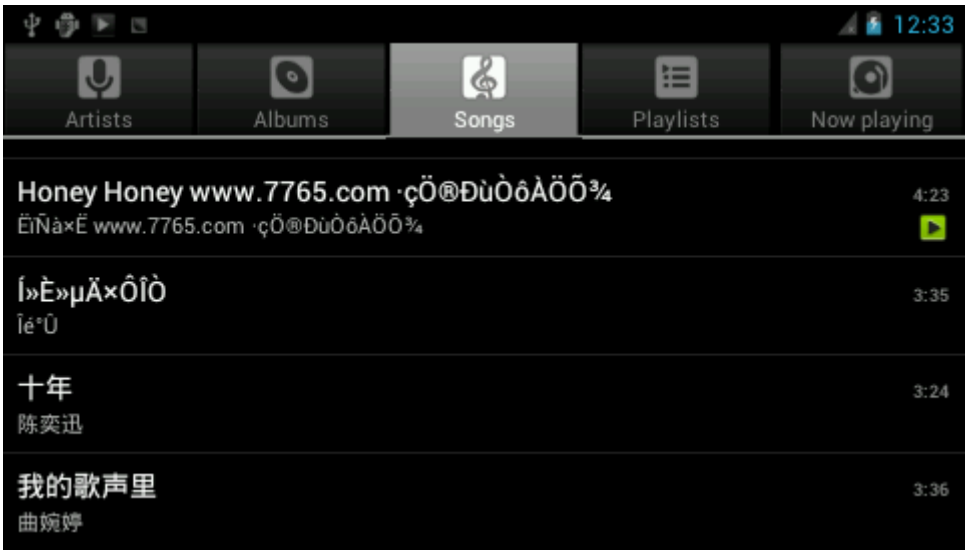


Figure 5-19

(3) Select a song to play. As shown in figure 5-20:

Figure 5-20

5.5.4 Ethernet Test

Note: Connect board to router by cable

(1) Enter interface and select icon "Ethernet",Refer to figure 5-21:

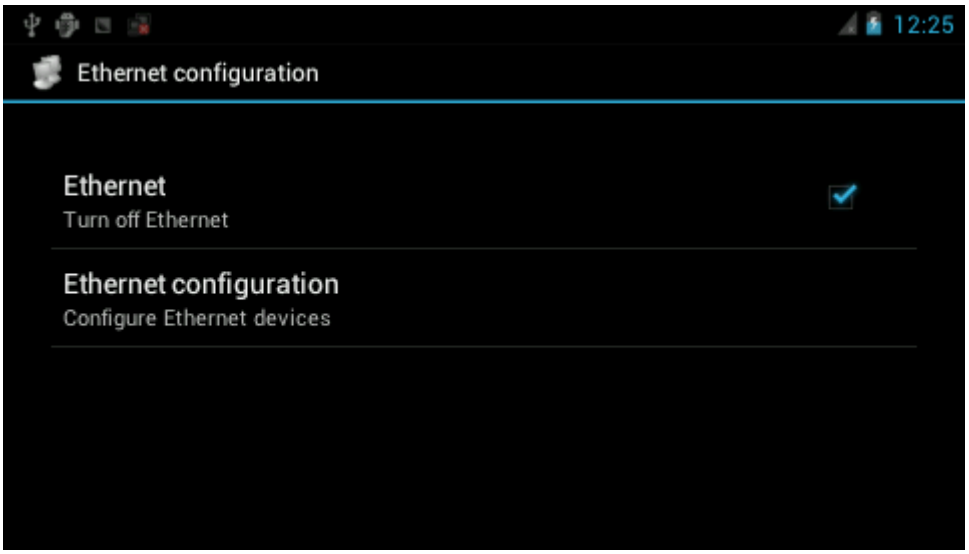


Figure 5-21

(2) Configure Ethernet, Select "Dhcp" to obtain dynamic IP. Otherwise, select "Static IP" manually to set IP address, subnet mask, DNS server, default gateway.. Refer to figure 5-22:

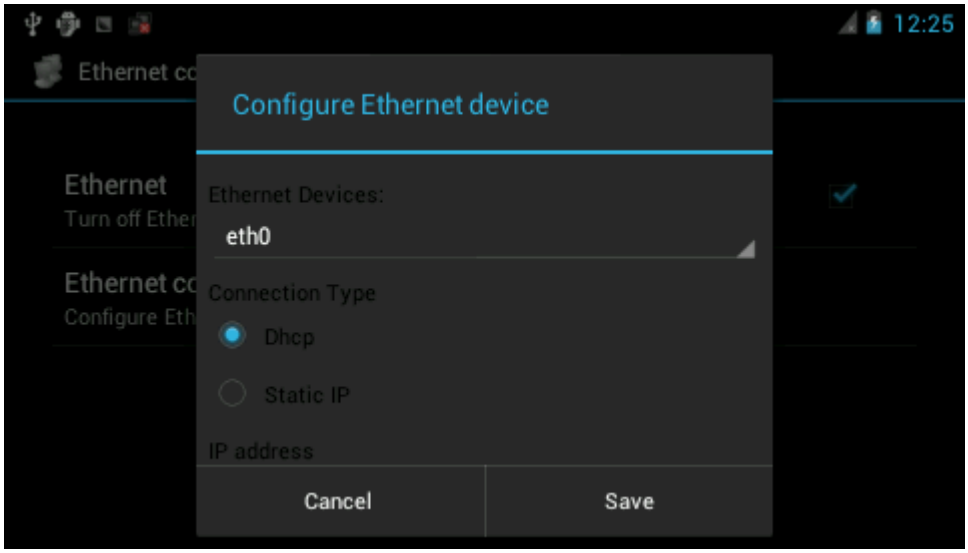


Figure 5-22

(3) After configure, input string: www.baidu.com. Refer to figure 5-23 figure 5-24:

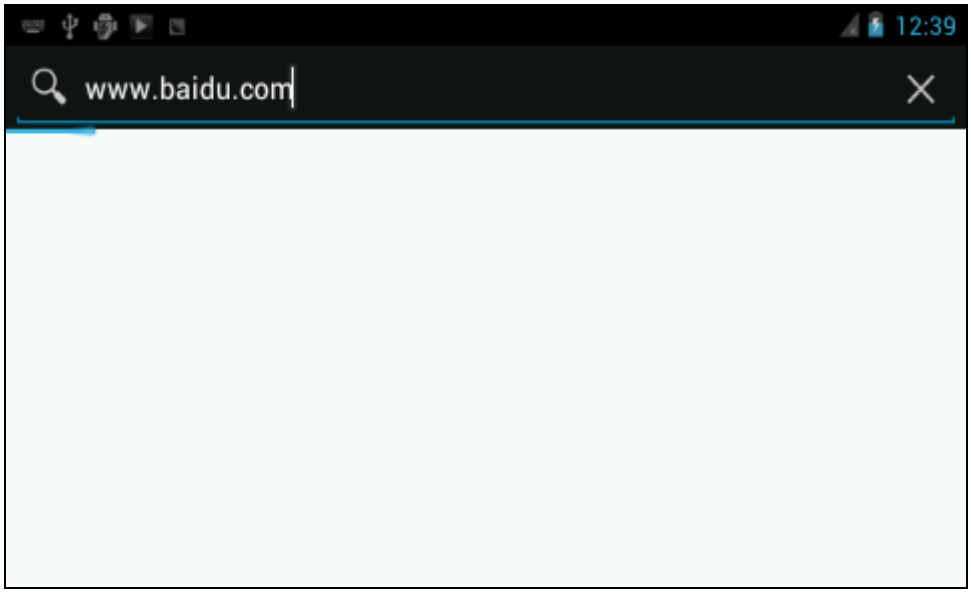


Figure 5-23

Figure 5-24

5.5.5 APK Test

The contents of this chapter, please refer to **" Android development environment and use Eclipse.pdf"**.

Appendix 1 FAQ

Q1: Report "Connection \USBserial\COMxx not found" (XX: port number, and according to the situation such as host machine is COM13, then XX is 13), pop up logfile file contents. As shown below:

```
-I- Waiting ...
-I- TCL platform : Windows NT
-I- SAM-BA 2.12 on : windows
-I- Retrieved arguments from command line :
-I- argv 0 : \usb\ARM0
-I- argv 1 : at91sama5d3x-ek
-I- argv 2 : sama5d3xek_demo_linux_nandflash.tcl
-E- Connection \USBserial\COM13 not found
-E- Connection list : COM1
```

Analysis and Answers:

This problem occurs because samba connection cannot be found "\ USBserial \ COMxx", ① Development board switch to 5 v (power off-position) ② Through the micro-USB line connected to the PC and development board ③ Connect the CPU module jumper JP1, JP2 backboard disconnect, hold CS_BOOT button At the same time development board switches to USB_5v ress NRST reset board , then turn SW2 on, and then start download.

Display logfile file as follows:

```
-I- Waiting ...
-I- TCL platform : Windows NT
-I- SAM-BA 2.12    on : windows
-I- Retrieved arguments from command line :
-I- argv 0 : \usb\ARM0
-I- argv 1 : at91sama5d3x-ek
```

```
-I- argv 2 : sama5d3xek_demo_linux_nandflash.tcl
-E- Connection \USBserial\COM3 not found
-E- Connection list : {\USBserial\COM13} COM1
```

It indicates that port isn't right, as above -I - argv 0: \ USBserial \ COM3 " , but connections list is "-E-Connection list: {\ USBserial \ COM13} COM1 " which shows native port is COM13 not COM3 and it needs to modify COM port (Note: COM port is that your host use).

Q2: Download system successfully, but can not start system.

Analysis and Answers:

If insert SD card to board, system may not start. For example, download LINUX system, if insert SD card to board, HyperTerminal display the following error message. Pulling out SD card can resolves the problem.

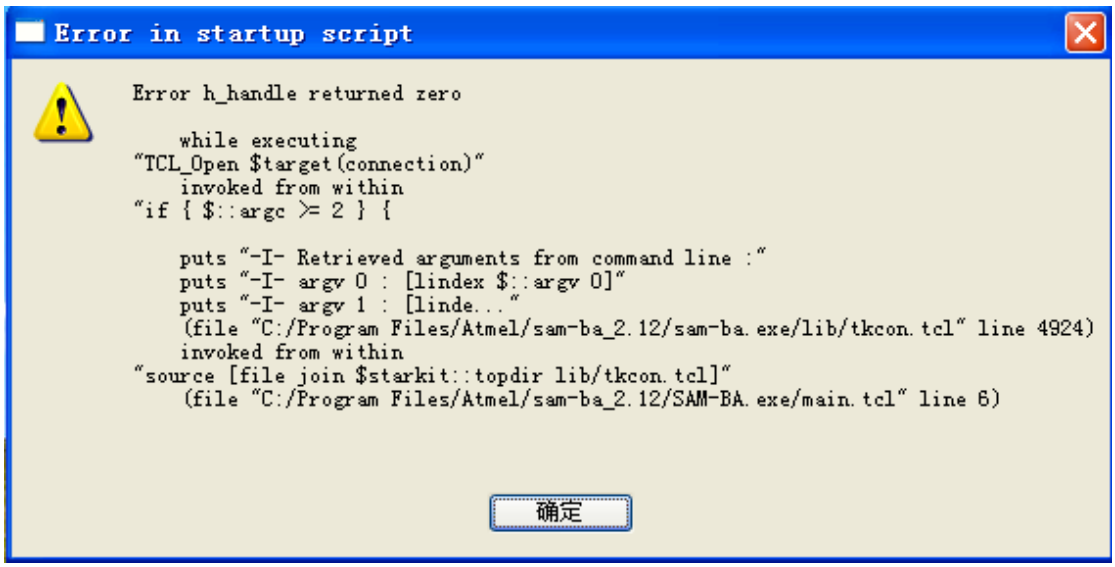
```
NAND read: device 0 offset 0x200000, size 0x250000
atmel_nand : one bit error on data. (data byte : b8, in page offset : 368, bit offset : 0x5)
atmel_nand : error corrected
atmel_nand : one bit error on data. (data byte : 71, in page offset : 1, bit offset : 0x5)
atmel_nand : error corrected
atmel_nand : one bit error on data. (data byte : f9, in page offset : 160, bit offset : 0x5)
atmel_nand : error corrected
atmel_nand : one bit error on data. (data byte : 60, in page offset : 437, bit offset : 0x5)
atmel_nand : error corrected
atmel_nand : one bit error on data. (data byte : 74, in page offset : 436, bit offset : 0x5)
atmel_nand : error corrected
atmel_nand : one bit error on data. (data byte : a0, in page offset : 340, bit offset : 0x5)
atmel_nand : error corrected
 2424832 bytes read: OK
## Booting kernel from Legacy Image at 22000000 ...
Bad Header Checksum
ERROR: can't get kernel image!
U-Boot>
```

Q4: Automatic download for a long time, HyperTerminal did not continue to output download information.

Analysis and Answers:

This may be stuck in automatic download process. It can end sam-ba.exe process in task manager and then restart download.

If start SAM-BA v2.11, Click Connect and pops up the following window when in automatic download:



It may be another SAM-BA is running. Stop SAM-BA in task manager, and then download it again.

Appendix 2 sales FAQ and technical support

How to buy

We accept paypal payment and bank wire transfer

1.Paypal payment

Please select the products add into shopping cart, the checkout web page will redirect to paypal.com for you payment. Shipment fee will calculated automatically by your location region.

2.Bank wire transfer

Please email or fax us with products list you want, we will send you a pro-invoice with order value total, shipping cost and bank information.

Shipping details

Please select the shipping area catalogue for you location. If you have carrier account to pay the shipment fee, please select "Freight collect" and email us the carrier account.

Please visit http://www.myirtech.com/support.asp for more details

Noted

1.The shipment will start in 3 biz days by Fedex Express, it usually take 7 days to reach regular cities or regions.

2.We will use DHL Express for West asia or middle east countries, it usually take 7 days to reach regular cities or regions.

3.The remote regions defined by Fedex/DHL may cause delay, 14 days in generally.

4.Some countries have strict import policy, we will help to make shipping invoice with you requirement, like invoice value, trade term, custom statements and H.S code etc. Please contact us with these shipment requirements if your country has strict custom affairs.

Support and maintains

MYIR provides 12 months warranty for hardware products if the defects or failures were not caused by wrong use.

Return steps for defective products

1. Please email or call us get a Return Merchandise Authorization (RMA) by providing purchase details and reasons for return (defective, incorrect etc).
2. MYIR will make a shipping invoice (list value total, item description etc) for you return request. China have strict limit on return products, so please use MYIR's shipping invoice to return items to avoid custom delay.

Contact:

Tel:+86-0755-22929657 Fax: +86-0755-2553 2724

Mail to: sales@myirtech.com support@myirtech.com

Website: www.myirtech.com